www.sciencemag.org/content/345/6198/799/suppl/DC1

Supplementary Materials for

A single-stranded architecture for cotranscriptional folding of RNA nanostructures

Cody Geary, Paul W. K. Rothemund, Ebbe S. Andersen*

*Corresponding author. E-mail: esa@inano.au.dk

Published 15 August 2014, *Science* **345**, 799 (2014)
DOI: 10.1126/science.1253920

This PDF file includes:

Contents

Materials and Methods

RNA Structural Modeling

A-form helices were generated in Assemble2 (*35*) using standard parameters for A-form RNA helices (*34*). Three-dimensional atomic models of RNA tiles were then constructed by joining helices with RNA motifs extracted from the RCSB PDB database (see Figure S3 for a detailed schematic of this process). The 180° kissing loop motif of the HIV-1 DIS (*31*) was extracted from the crystal structure (PDB_ID: 1B8R), and the 120° loop-loop complex of RNA i/ii inverse loop (*32*) from the NMR solution structure (PDB_ID: 2BJ2). The tetraloops GNRA and UNCG were extracted from the ribosomal structure (PDB_ID: 2AVY). The merging of RNA fragments was performed using automated functions built into the Assemble2 software, and the final structures were then refined using a recursive geometric refinement function in Assemble2 (*35*). All molecular models in this work were rendered using UCSF Chimera (*36*).

RNA Sequence Design

RNA secondary structures annotated with stem and loop lengths were generated automatically from three-dimensional RNA models using RNAView (distributed with Assemble). From these structures, text-based secondary structure designs were built by hand. The sequences of the 180° programmable kissing loops and 120° programmable kissing loops were chosen as previously reported (*8, 13*) and added to the designs. The 5' end of each sequence was constrained to begin with GGAA, an optimal leader sequence for T7 RNA polymerase. Some base pairs in dovetail crossovers and adjacent base pairs were constrained to be strongly stacking G-C pairs in an attempt to specify which conformation of the junction would be preferred. Sequences were further constrained to contain at least one GU wobble pair per every eight continuous base pairs in order to avoid secondary structure in the RNA-encoding DNA template and simplify its synthesis; such positions were specified by using the letter 'K'. Text-based secondary structure designs with sequence constraints were run through a Perl script (see SI Note 2 for source code) that converted them to the dot-bracket notation and sequence constraints appropriate for NUPACK (*40*) and wrote out NUPACK design files (Fig. S5). For GU-wobble positions specified with a 'K', NUPACK assigns either a G or a U and thus either G-U or U-G pairs are incorporated at such a position.

DNA Template Synthesis

DNA templates for all RNA designs were ordered as "custom gBlock" double-stranded DNA oligos from Integrated DNA Technologies (IDT). Custom gBlocks were supplied as 200 ng of lyophilized DNA. DNA gBlocks were reconstituted in TE buffer to a concentration of 4 ng/µl and were amplified by PCR over 30-35 cycles using primers specific to the 5' and 3' ends of the template. PCR amplified DNAs were purified using a standard Qiagen DNA purification kit and suspended in TE buffer. Primers used for PCR amplification are listed in SI Note 1.

Preparation of purified RNA tiles

RNA tiles for incubation and annealing experiments were transcribed and purified individually. ~25 ng/µl of template DNA was mixed in a transcription buffer containing

15 mM $Mg(OAc)_2$, 40 mM Na-OAc, 50 mM Tris-OAc and 0.1% Tween20. NTPs (2.5 mM each) and DTT (1 mM) were added prior to the addition of T7 RNA polymerase (~1 U/50 µl). Reactions were carried out in 50 µl volumes at 37.0°C for 1.5 hours, and completed by the addition of 2 U/50 µl DNase and incubated for an additional 30 minutes at 37.0°C. Products were purified on a 6% Acrylamide:bis (19:1) gel containing 8 M Urea and 1X TE buffer. The desired RNAs were cut out of the gel and eluted in 200 nM NaCl, 10 mM Tris (pH 7.5), 0.5 mM EDTA overnight. RNAs were precipitated with a 2-3 volume excess of EtOH (90%) and dried under vacuum.

Four different RNA assembly protocols were used in this work (see S8 for graphical depiction):

1. Solution incubation

Freshly-cleaved mica affixed to a metal disk, with a 50 µl drop of AFM buffer was preheated to 37.0°C and set aside. Purified RNA tiles were diluted in pure water at a concentration calibrated to give coverage, but not large aggregates, on the mica surface (~200 nM for 2H tiles, ~25 nM for 4H tiles and ~20 nM for 6H tiles). Tiles were next subjected to heat denaturation/renaturation (90°C for 3 mins, 4°C for 5 mins) prior to the experiment. Tiles were allowed to come to room temperature, and 5X AFM buffer was added. The sample was next incubated at 37.0°C for 30 minutes. After incubation, 6 µl of RNA sample was added to the preheated 50 µl buffer drop on the mica, and imaged immediately.

2. Cotranscriptional assembly (Cotranscriptional Protocol, Figure 1F)

Freshly-cleaved mica affixed to a metal disk was preheated to 37.0°C and set aside. RNAs were cotranscriptionally assembled in one-pot reactions. All template DNAs for a given lattice design were mixed together (2 ng/µl each) in a reaction buffer containing 6 mM $Mg(OAc)_2$, 40 mM Na-OAc, 40 mM KCl, and 50 mM Tris-OAc (pH 7.8). NTPs (0.5 mM each) and DTT (1 mM) were added prior to the addition of T7 RNA polymerase (~0.1 U/50 µl). Reactions were carried out in 50 µl volumes at 37.0°C for 10 minutes. For AFM imaging the completed reactions were diluted by a factor of 5X in AFM buffer, 50 µl of solution was deposited onto the 37.0°C mica, and imaged immediately. For PAGE experiments the reactions were mixed with 1 U/50 µl of DNase for 15 minutes to remove the template DNA prior to loading the products on the gel.

3. Mica-annealing (Mica-annealing Protocol, Figure 1E)

Freshly-cleaved mica affixed to a metal disk was placed inside a covered 50 mm glass petri dish containing a wet kimwipe (to maintain the humidity in the chamber). The chamber was then pre-heated to 45°C on an aluminum block incubator. A 50 µl drop of AFM buffer was added to the mica and pre-heated for 5 mins). Purified RNA tiles were diluted in pure water at a concentration calibrated to give coverage, but not large aggregates, on the mica surface (~200 nM for 2H tiles, ~25 nM for 4H tiles and ~20 nM for 6H tiles). Tiles were next subjected to heat denaturation/renaturation (90°C for 3 mins, 4°C for 5 mins, buffer added at RT) prior to the experiment. 6 µl of RNA sample was added to the mica on top of the pre-heated (45°C) buffer drop. The heating block was

then turned off and allowed to cool slowly to 30°C (over ~1.5 h). Samples were removed and imaged immediately at room temperature.

4. Transcription directly on mica (Transcription on Mica Protocol, Figure 4A)

Freshly-cleaved mica affixed to a metal disk was placed inside a covered 50 mm glass petri dish containing a wet kimwipe (to maintain the humidity in the chamber). The chamber was then pre-heated to 45°C on an aluminum block incubator. 25 µl of AFM buffer was added to each mica puck and pre-heated for 5 mins). Template DNA for the 2H-AE single tile system was diluted (2 ng/µl) in a reaction buffer containing 6 mM $Mg(OAc)_2$, 40 mM Na-OAc, 40 mM KCl, and 50 mM Tris-OAc (pH 7.8). NTPs (0.5 mM each) and DTT (1 mM) were added prior to the addition of T7 RNA polymerase (~0.1 U/50 µl). 25 µl of this reaction was then immediately mixed into the pre-warmed 45°C buffer on the mica surface. The heating block was then turned off and allowed to cool slowly to 30°C (over ~1.5 h). Samples were removed and imaged immediately at room temperature.

AFM Buffer

Tris-Borate 1X (pH 8.1), 2 mM $Mg(OAc)_2$, 50 mM KCl, 50 mM NaCl.

AFM imaging

AFM images were collected in tapping mode under buffer using a Digital Instruments Multimode AFM with a Nanoscope IIIA controller and either an E or J-scanner. Olympus TR400PSA silicon nitride probes with a spring constant of ~0.08 N/m were typically used for imaging, with a drive frequency of ~6-9 kHz.

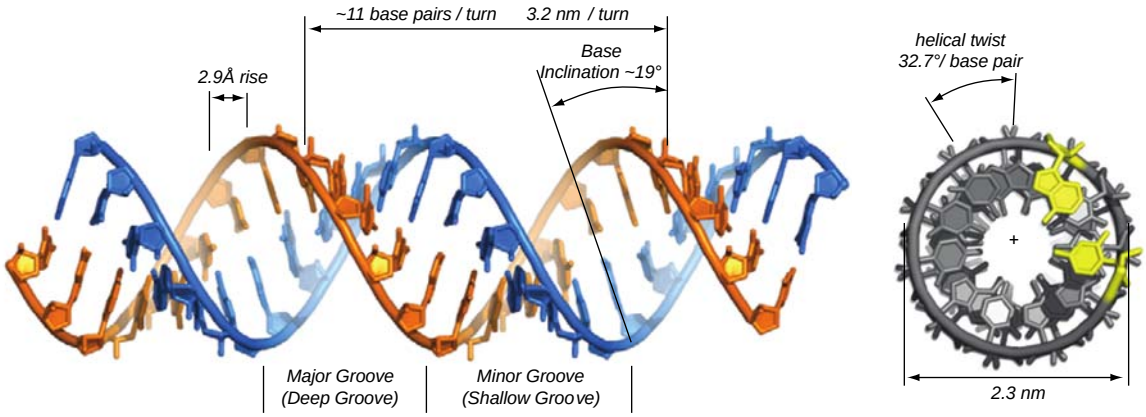


Fig. S1. Schematic of an A-form RNA helix.

RNA is characterized by a steep base-pair inclination of 19° relative to the helical axis, and a helicity of 11 bp/turn (*34*). The major and minor grooves of RNA are typically referred to instead as the deep and shallow grooves, respectively, due to the fact that the minor groove is wide and shallow compared to the major groove. Another characteristic of RNA is that from the top view the helix has a hollow central cavity, which is in contrast to DNA for which the base pairs pass through the central axis.

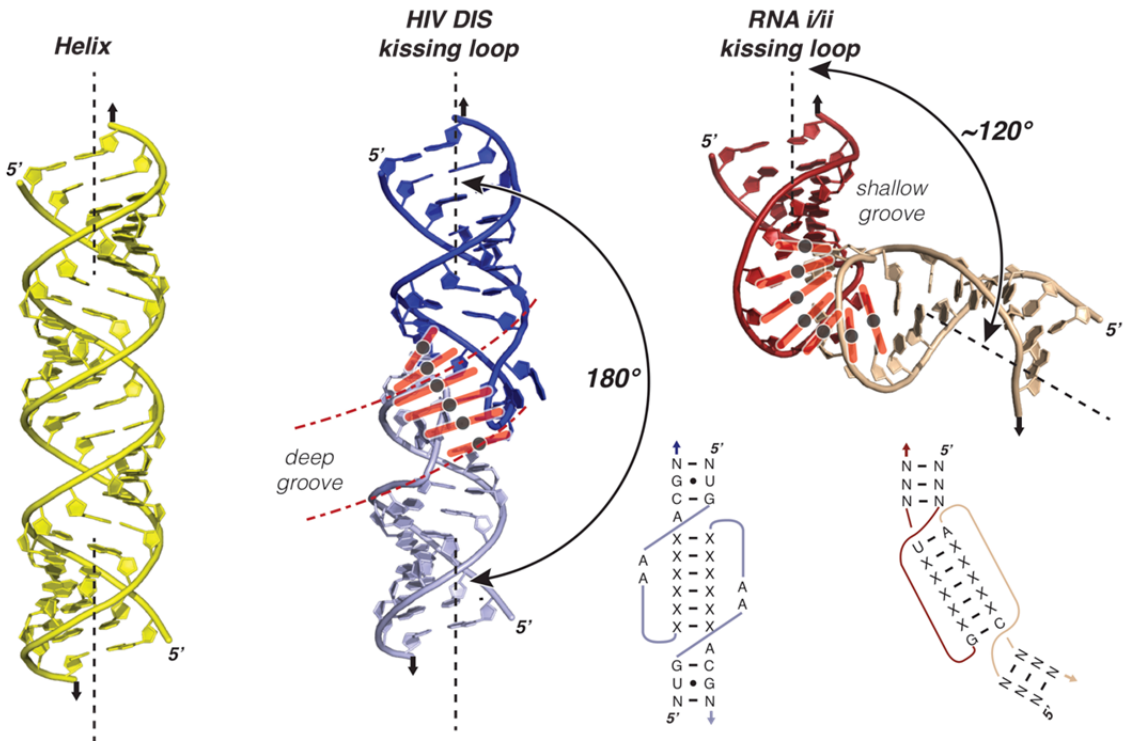


Fig. S2. Comparison between A-form helix and kissing-loop motifs.

Alignment of a typical A-form helix with the 180° and 120° kissing loops (KLs). The 180° KL (blue) originates from the DIS dimerization domain of HIV RNA (*31*), and forms 6 base pairs between two loops resulting in a coaxial stack. Note that the blue helix is perfectly in phase with the normal A-form helix of RNA, thus the 180°KL motif can be substituted into a helix without changing the shape of the helix significantly. The 120° KL (red) originates from the dimerization domain of the ColEI plasmid from *Escherichia coli (32)*, and forms 7 base pairs between the loops resulting in a continuous, but curved, coaxial stack. In the 180° KL the two consecutive unpaired bases span the narrow deep groove of the RNA helix. By contrast, in the 120°KL the lack of unpaired bases to cross the deep groove may explain how the motif induces bending in the shallow groove.

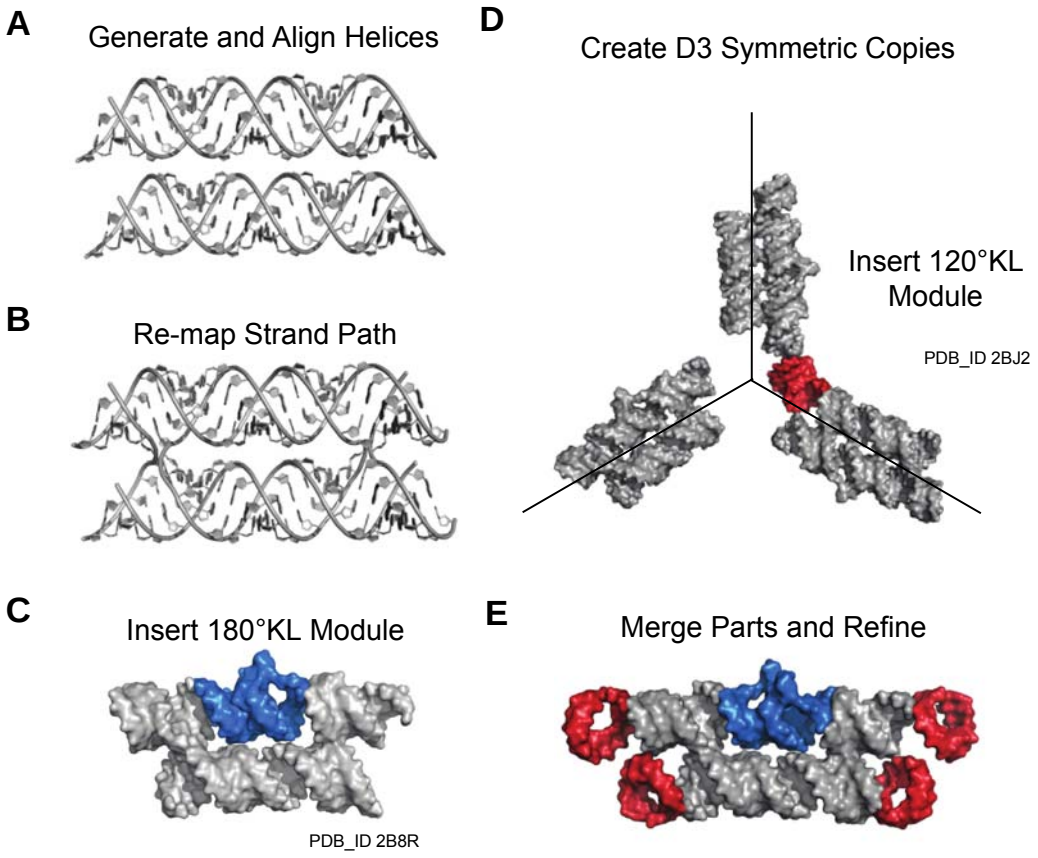


Fig. S3. RNA design method.

Stepwise procedure for designing modular RNA structures. (A) Helices are generated in Assemble2 (*35*) from classic A-form helix parameters (*34*). (B) The nucleotides in the model are re-numbered based on the desired new strand path, and the coordinate set is refined in Assemble2 using a recursive geometric refinement function (*35*). At step (C) and (D) the inserted structural module is highlighted in blue and red, respectively. Step (E) shows the final RNA structure after refinement in Assemble2, with KL modules highlighted.

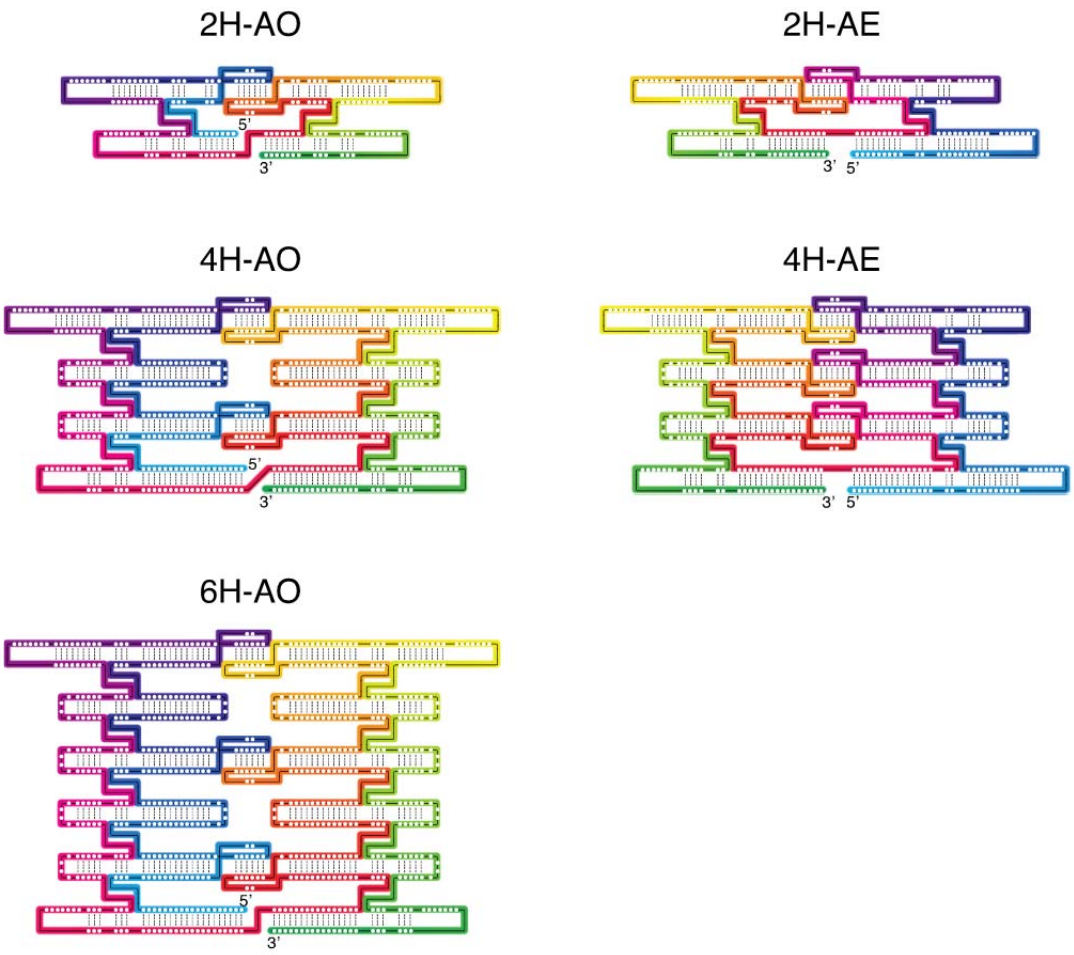


Fig. S4. Strand-path diagrams.

Strand-path diagrams showing the order of synthesis, 5’ to 3’, from blue to purple to red to orange to yellow to green.

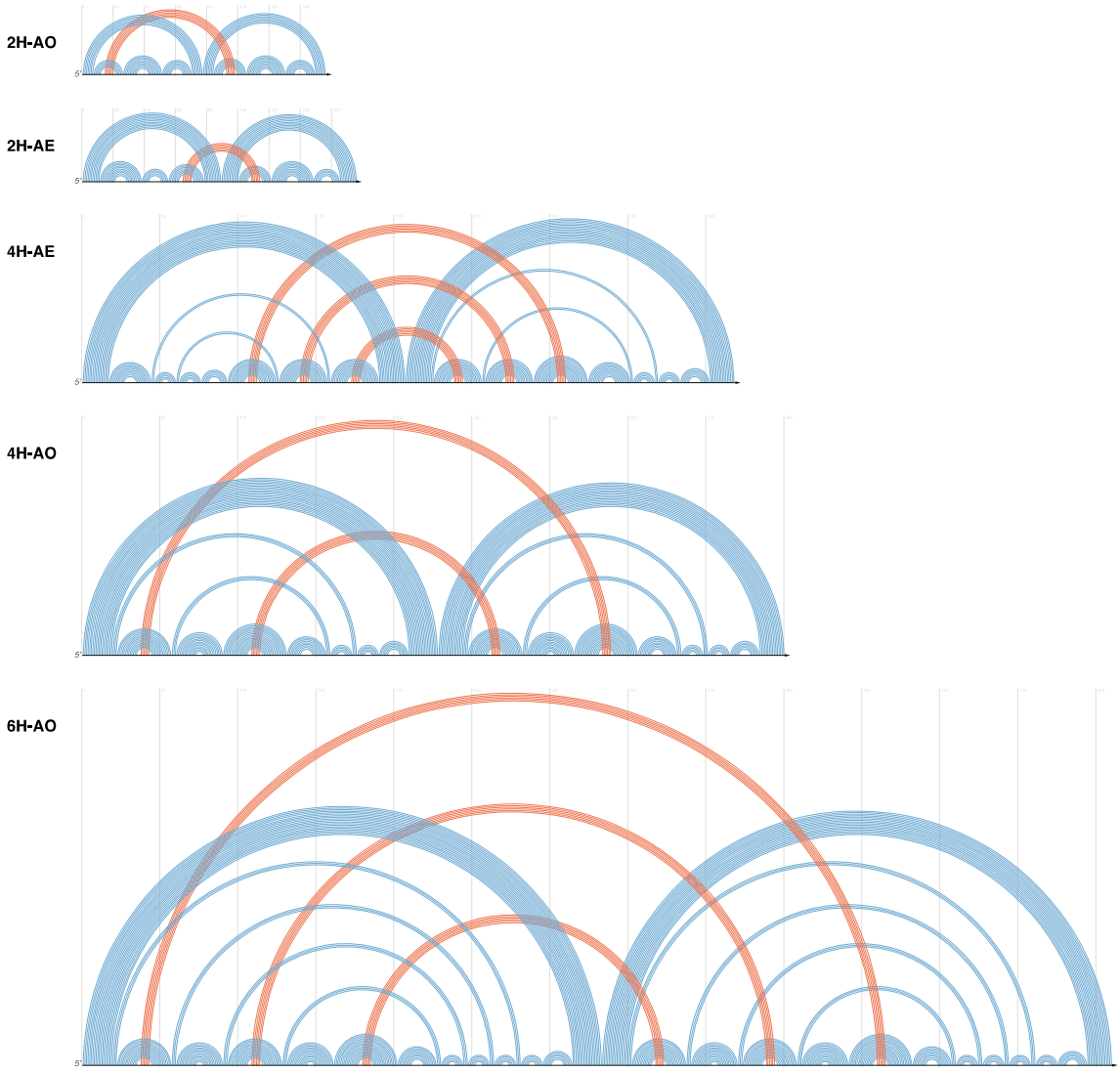
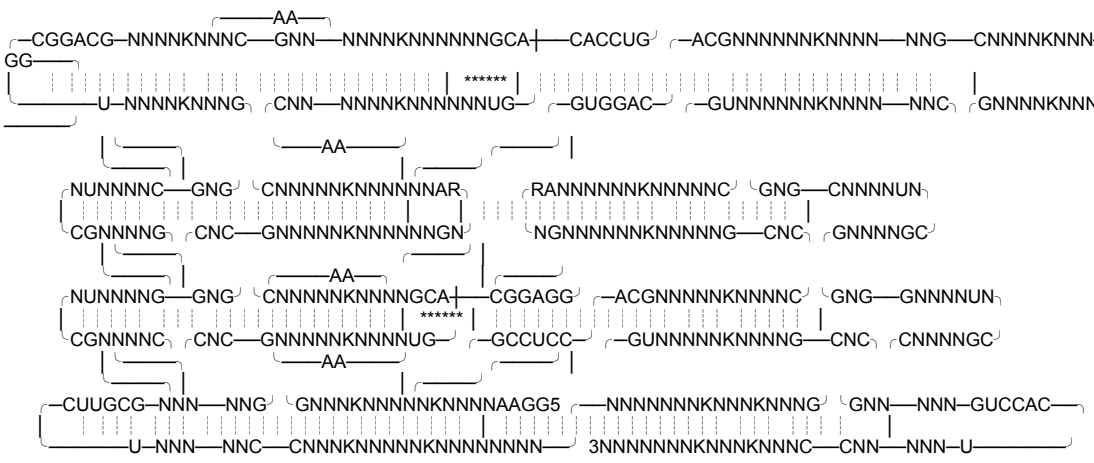


Fig. S5. Arc diagrams showing the pseudoknotted structure of tiles.

Arc diagrams were generated with RChie (e-rna.org). Each black line represents an RNA strand from left to right, 5’ to 3’. The blue represent base pairs in the secondary structure and the red arcs represent base pairs in the tertiary structure. Blue arcs indicate base pairs in helical domains and dovetail seams. Red arcs indicate base pairs within the 180° KL interactions. Because blue and red arcs cross, all these structure are said to be pseudoknotted. For each tile design the structure can be divided into two domains, where the largest blue arcs meet in the middle. The two independent secondary structural domains are bridged by the KL interactions. When the RNA folds, it is expected that strong blue interactions would form a non-pseudoknotted structure first, which then later become a pseudoknot upon formation of the red KL interactions.

Input file:



The 5p end is found at row 17, column 44 and is running left.
There are 450 nucleotides in the blueprint file.
The structure has been successfully traced (3p end found).
The are 450 nts from 5p to 3p.

Output: The sequence and structure

GGAANNNNKNNNNNKNNNGCNCGNNNNNKNNNNUGAAGGAGGCACGNNNNKNNNNNCCNCGNNNNNKNNNNNNGNRANNNNNNKN
NNNNCCNNNNNNKNNNNNNUGAAGUCCACACGNNNNNNKNNNNNNGCNNNKNNNNGCAGGCUNNNNKNNNGGNGCNNNNUNCGNN
NNGGNGGNNNNUNCGNNNNCGNNNNNGCGUUCUNNNNNCCNNNKNNNNNKNNNNNNNNNNNNNNNKNNNKNNNGCNCGNNNNKNN
NNNUGAAGCCUCCACGNNNNNKNNNNCCNCGNNNNNKNNNNNNGNRANNNNNNKNNNNNCCNNNNNNKNNNNNNUGAAGUGGACA
CGNNNNNNKNNNNNNGCNNNNKNNNGAUGGUUNNNKNNNNGGNGCNNNNUNCGNNNNGGNGGNNNNUNCGNNNNCGNNNNNGUC
CACUNNNNNCCNNNKNNNKNNNNNNN
((((((((((((((((((((((((((((((((..[[[[[[.)))))))))))))))))(((((((((((((((((....)))))))))))))))))(((((((((((((((((..[[[[[[.)))))))))))))))))(((((((((.....)))))))))(((((....)))))))((((....))))))(
((((((.....))))))))))))))))))))))))))))))(((((((((((((((((((((((((((((((..]]]]]].)))))))))))))))))(((((((((((((((((....)))))))))))))))))(((((((((((((((((..]]]]]].)))))))))))))))))(((((((((((...)))))))))))
))))(((((....)))))))((((....))))))(((((((.....)))))))))))))))))))))))))

Begin NuPack Code (copy below this line):
material = RNA
temperature[C] = 37.0
trials = 4
sodium[M] = 1.0
dangles = some
allowmismatch = true
structure RNA_seq =
((((((((((((((((((((((((((((((((.....)))))))))))))))))(((((((((((((((((....)))))))))))))))))(((((((((((((((((.....)))))))))))))))))(((((((((.....)))))))))(((((....)))))))((((....))))))(
((((((.....))))))))))))))))))))))))))))))(((((((((((((((((((((((((((((((.....)))))))))))))))))(((((((((((((((((....)))))))))))))))))(((((((((((((((((.....)))))))))))))))))(((((((((((...)))))))))))
))))(((((....)))))))((((....))))))(((((((.....)))))))))))))))))))))))))
domain a =
GGAANNNNKNNNNNKNNNGCNCGNNNNNKNNNNUGAAGGAGGCACGNNNNKNNNNNCCNCGNNNNNKNNNNNNGNRANNNNNNKN
NNNNCCNNNNNNKNNNNNNUGAAGUCCACACGNNNNNNKNNNNNNGCNNNKNNNNGCAGGCUNNNNKNNNGGNGCNNNNUNCGNN
NNGGNGGNNNNUNCGNNNNCGNNNNNGCGUUCUNNNNNCCNNNKNNNNNKNNNNNNNNNNNNNNNKNNNKNNNGCNCGNNNNKNN
NNNUGAAGCCUCCACGNNNNNKNNNNCCNCGNNNNNKNNNNNNGNRANNNNNNKNNNNNCCNNNNNNKNNNNNNUGAAGUGGACA
CGNNNNNNKNNNNNNGCNNNNKNNNGAUGGUUNNNKNNNNGGNGCNNNNUNCGNNNNGGNGGNNNNUNCGNNNNCGNNNNNGUC
CACUNNNNNCCNNNKNNNKNNNNNNN
RNA_seq.seq = a
prevent = AAAA, CCCC, GGGG, UUUU, KKKKKK, MMMMMM, RRRRRR, SSSSSS, WWWWWW, YYYYYY

Fig. S6. A sample NUPACK design session for 4H-AO.

At the top, the secondary structure design and sequence constraints are input in a human-readable format as ASCII text art. Our Perl script (see SI Note 2 for source code) detects the beginning of the strand starting at the “5”, which indicates the 5’ end, and generates a sequence constraint and secondary structure in dot-bracket notation. A NUPACK script to design the inputted RNA structure is then automatically generated at the end.

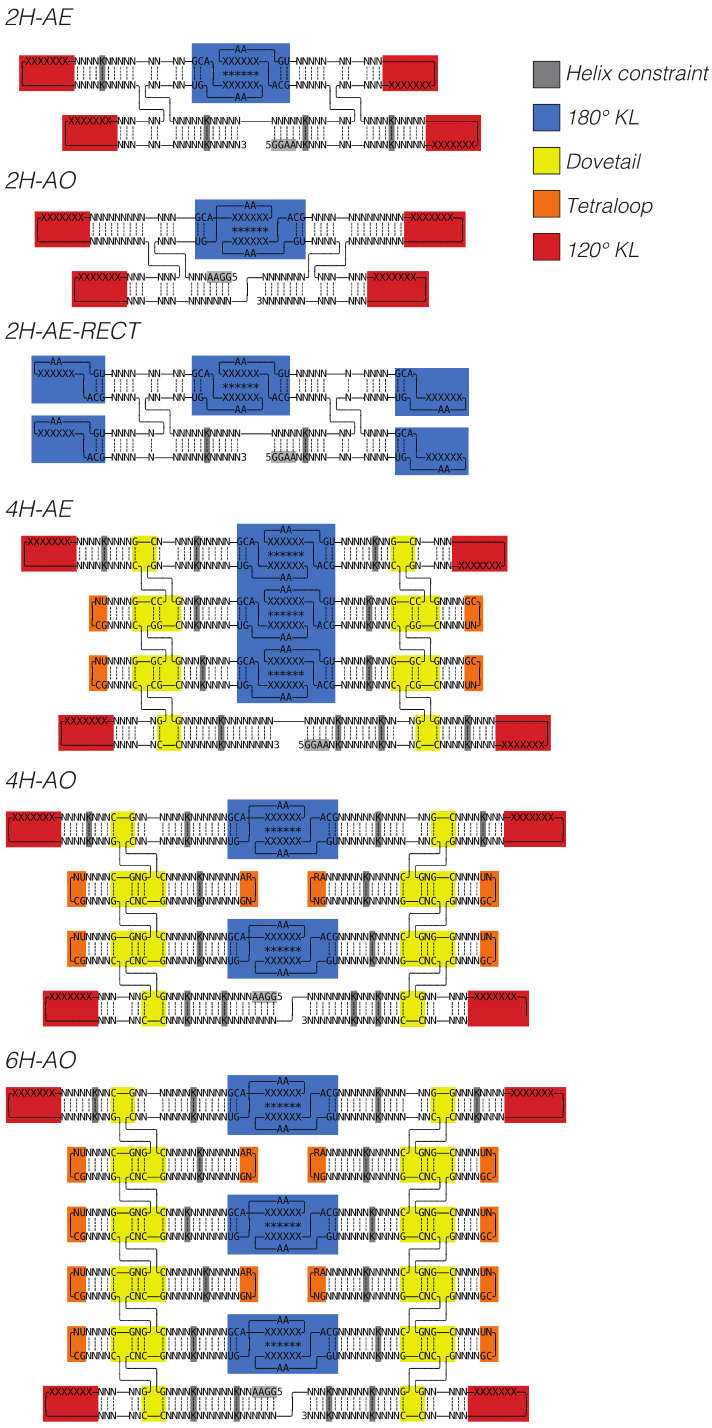


Fig. S7. Modular design.

Secondary structure diagrams showing the base pair and sequence constraints used for all NUPACK designs. Diagrams are colored to indicate conserved tertiary motifs. Gray regions indicate where sequence constraints have been added, for example to force the inclusion of a G-U wobble or the 4-nt leader that is optimal for T7 RNA polymerase.

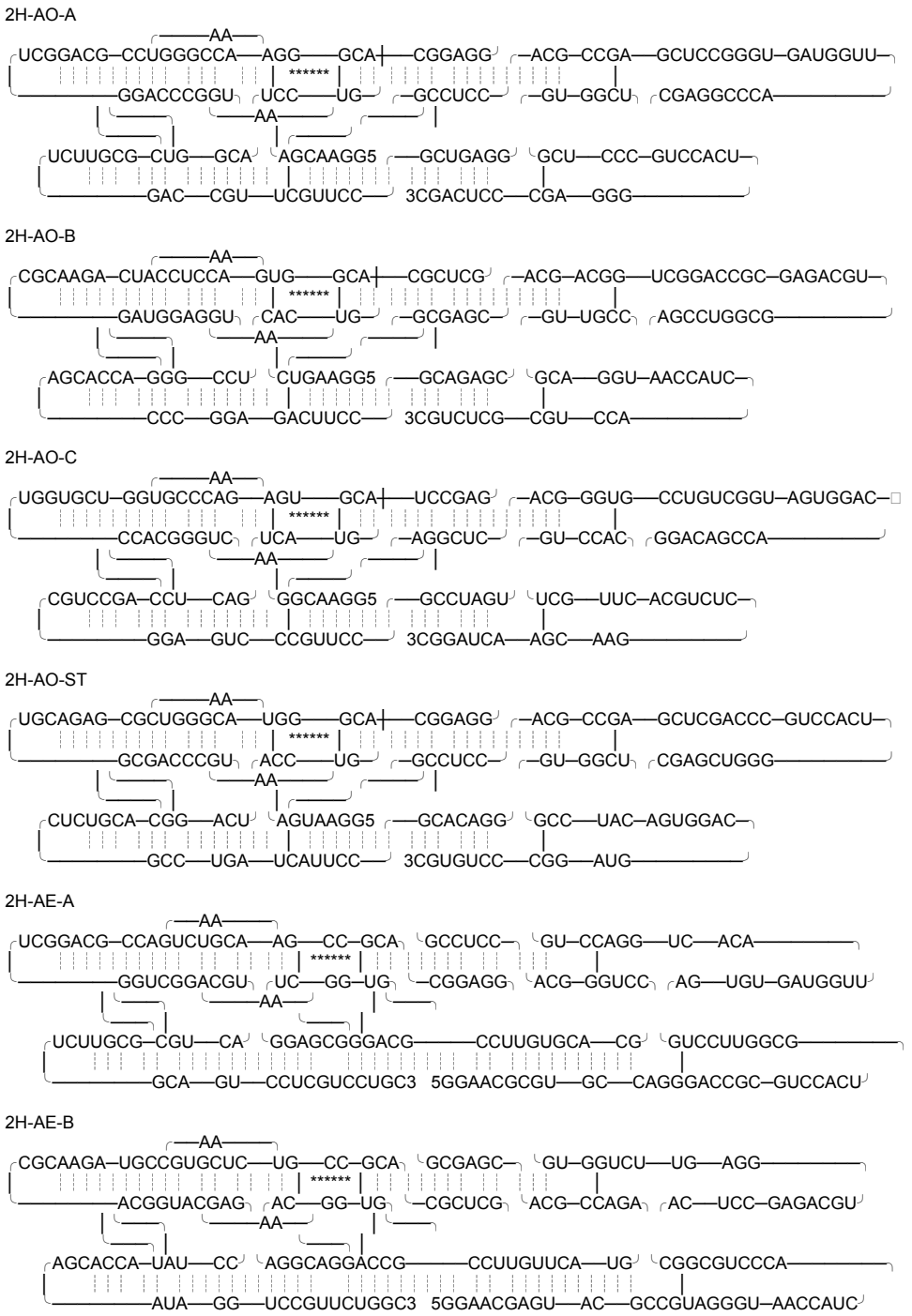


Fig. S8. Secondary structure diagrams.

Secondary structure diagrams of all RNAs used in this study. (*) represents KL interactions that are pseudoknotted. Three-tile systems have ‘A’, ‘B’, and ‘C’ tiles. Tiles for single tile systems are denoted ‘ST’ and for two-tile rectilinear systems, ‘RECT’. Figure continues on the following pages.

Fig. S8. (continued)

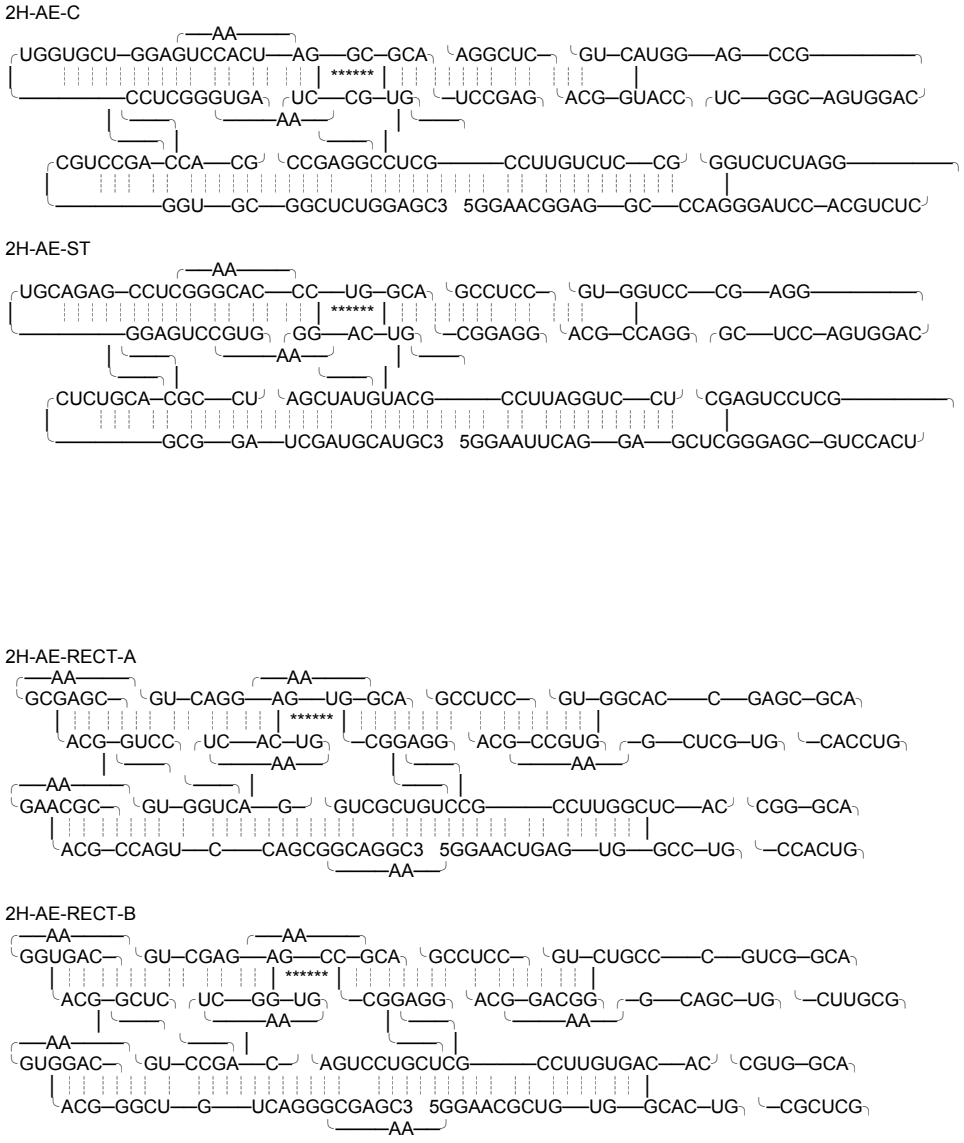


Fig. S8. (continued)

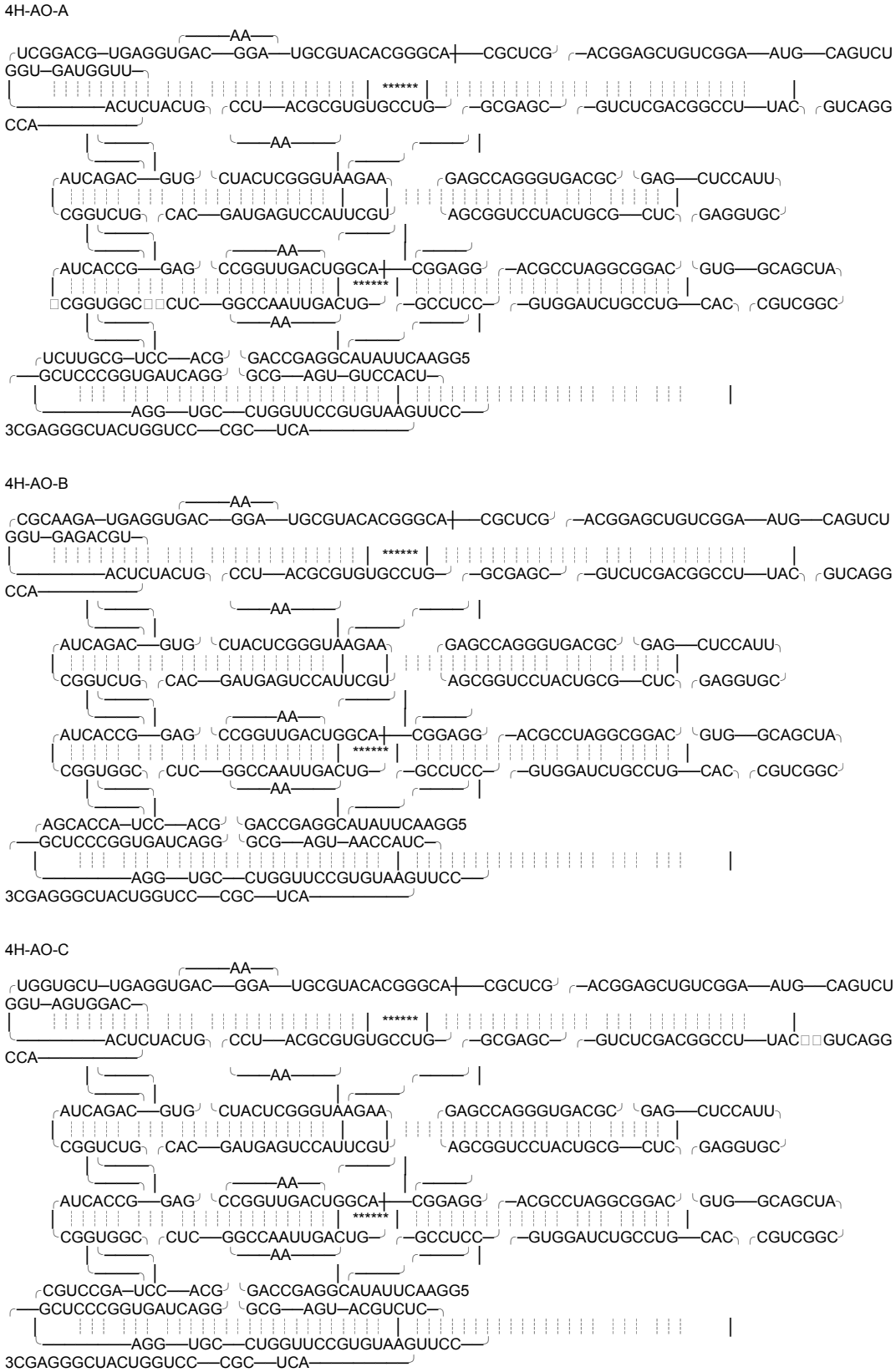
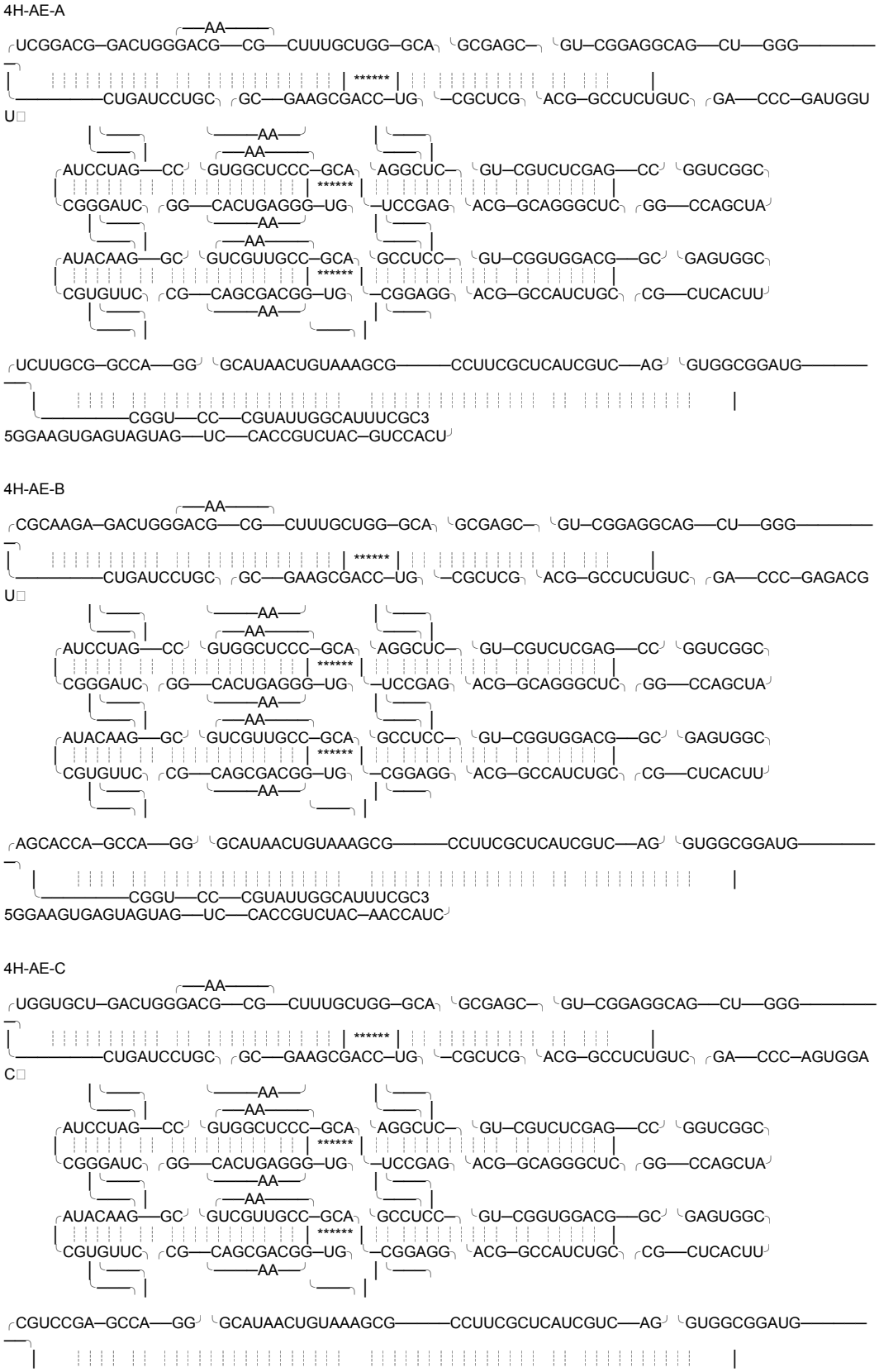


Fig. S8. (continued)



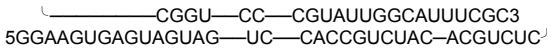
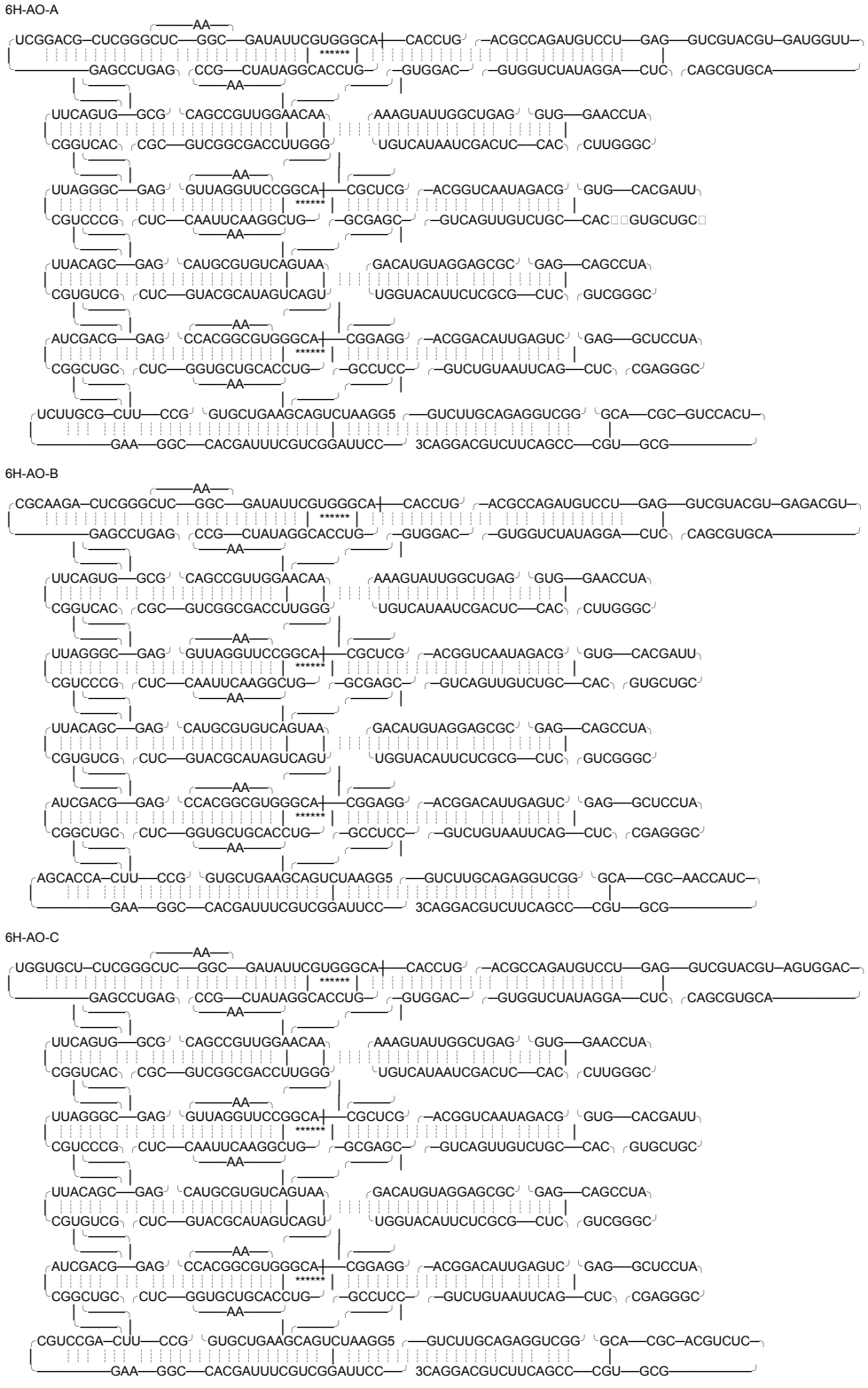

CGGU—CC—CGUAUUGGCAUUUCGC3
5GGAAGUGAGUAGUAG—UC—CACCGUCUAC–ACGUCUC

Fig. S8. (continued)



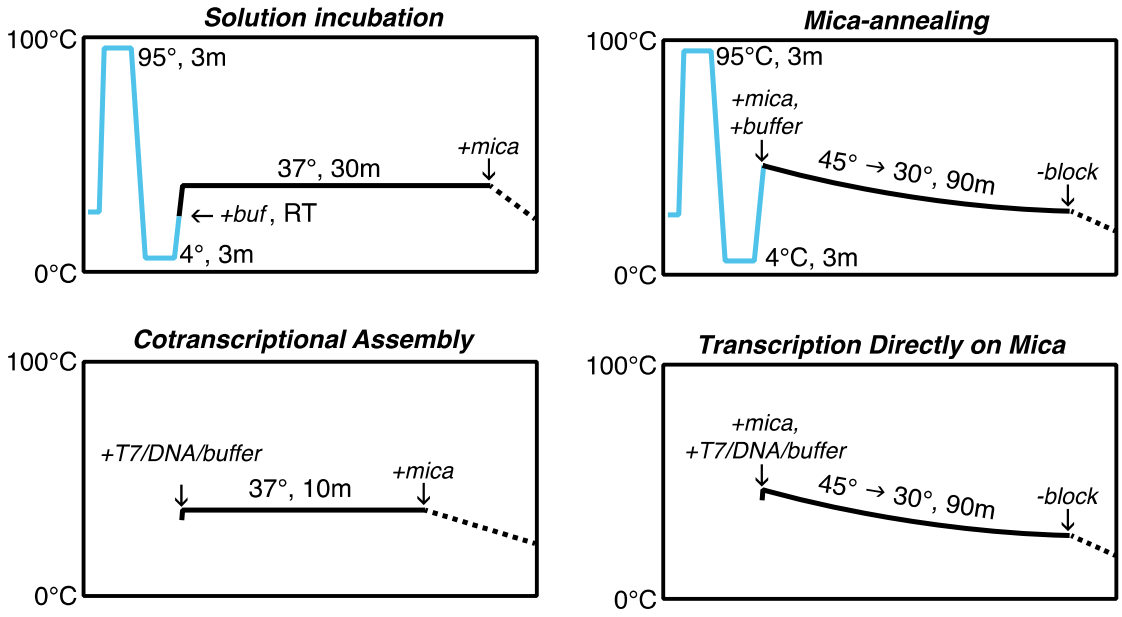


Fig. S9. Assembly protocols.

Summary of RNA assembly protocols. Blue lines represent our RNA denaturation/renaturation protocol, during which PAGE-purified RNAs were suspended in pure water, denatured at 95°C and then snap-cooled to 4°C. Buffer was added at room temperature before heating to 37°C (solution incubation) or 45°C (mica annealing). The dotted line begins at the moment the mica sample is removed from the heating block and is no longer under strict temperature control. The line trends down to show that the sample cools to room temperature (~20°C).

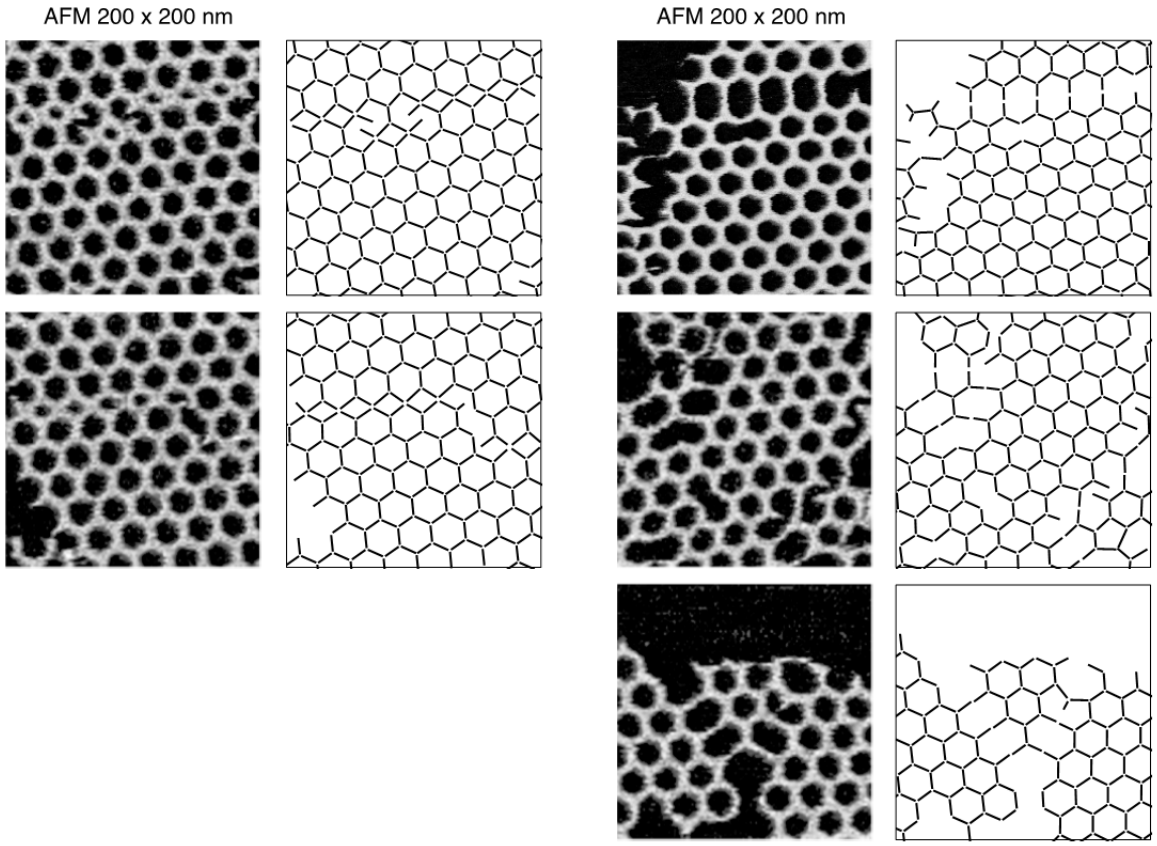


Fig. S10. Interpretation of AFM for 2H-AE single-tile lattices.

2H-AE single-tile lattices were prepared by mica annealing. To the right of each AFM image is an interpretation of the RNA tile connectivity.

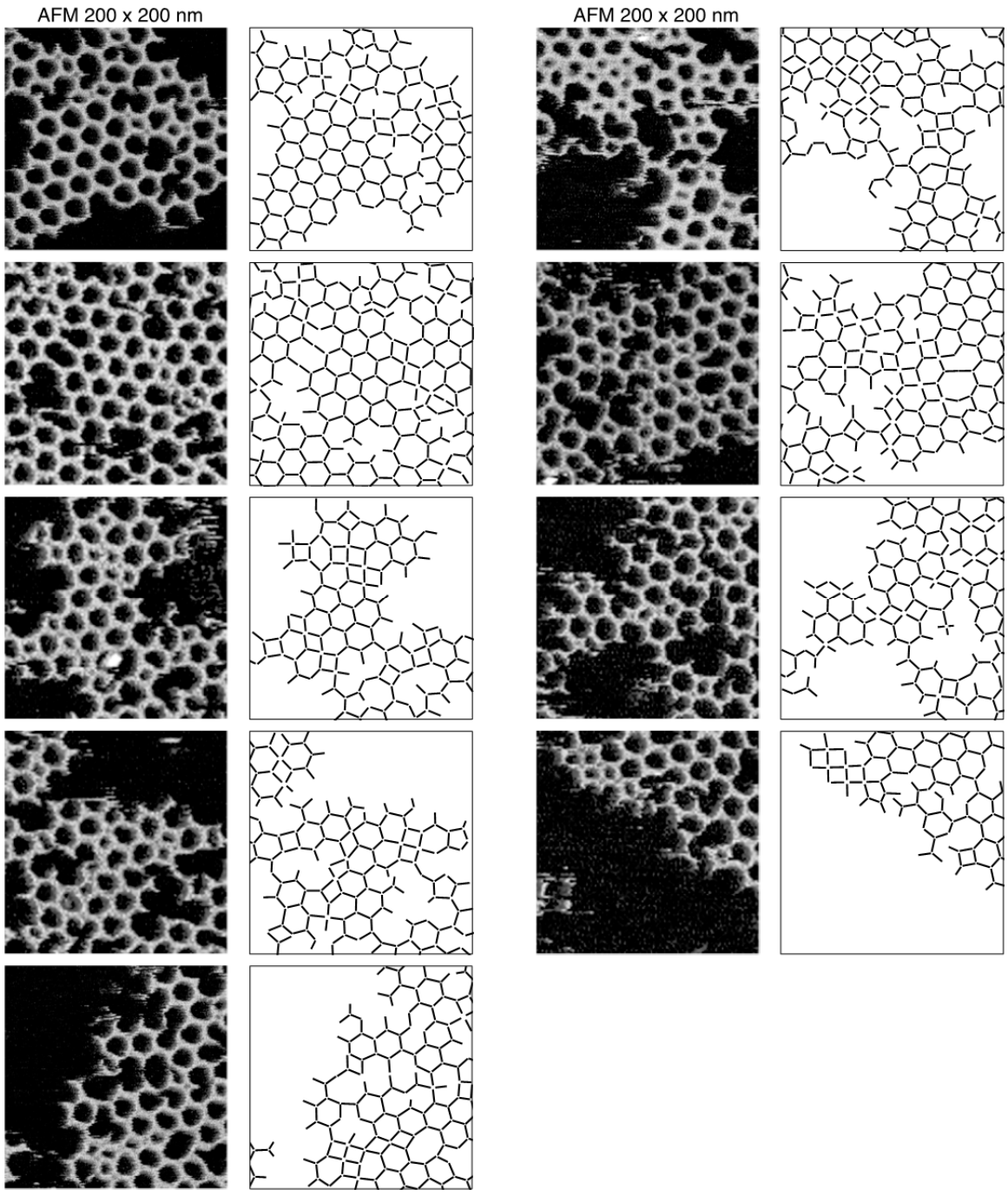


Fig. S11. Interpretation of AFM for 2H-AO single-tile lattices.

2H-AO single-tile lattices were prepared by mica-annealing. To the right of each AFM image is an interpretation of the RNA tile connectivity.

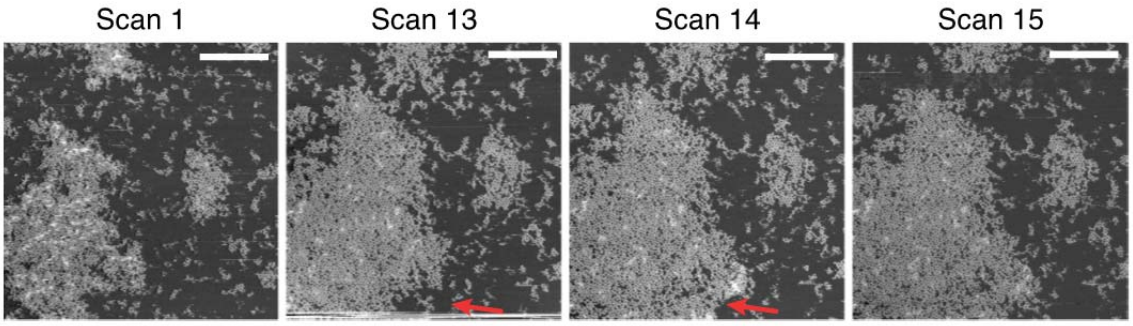


Fig. S12. Solution assembly and deposition of 2H-AO.

Successive AFM scans of 2H-AO three-tile hexagonal lattices that were assembled in solution before deposition on mica. The red arrows indicate where a new patch of tiles lands on the mica from the solution. Scale bar: 500 nm.

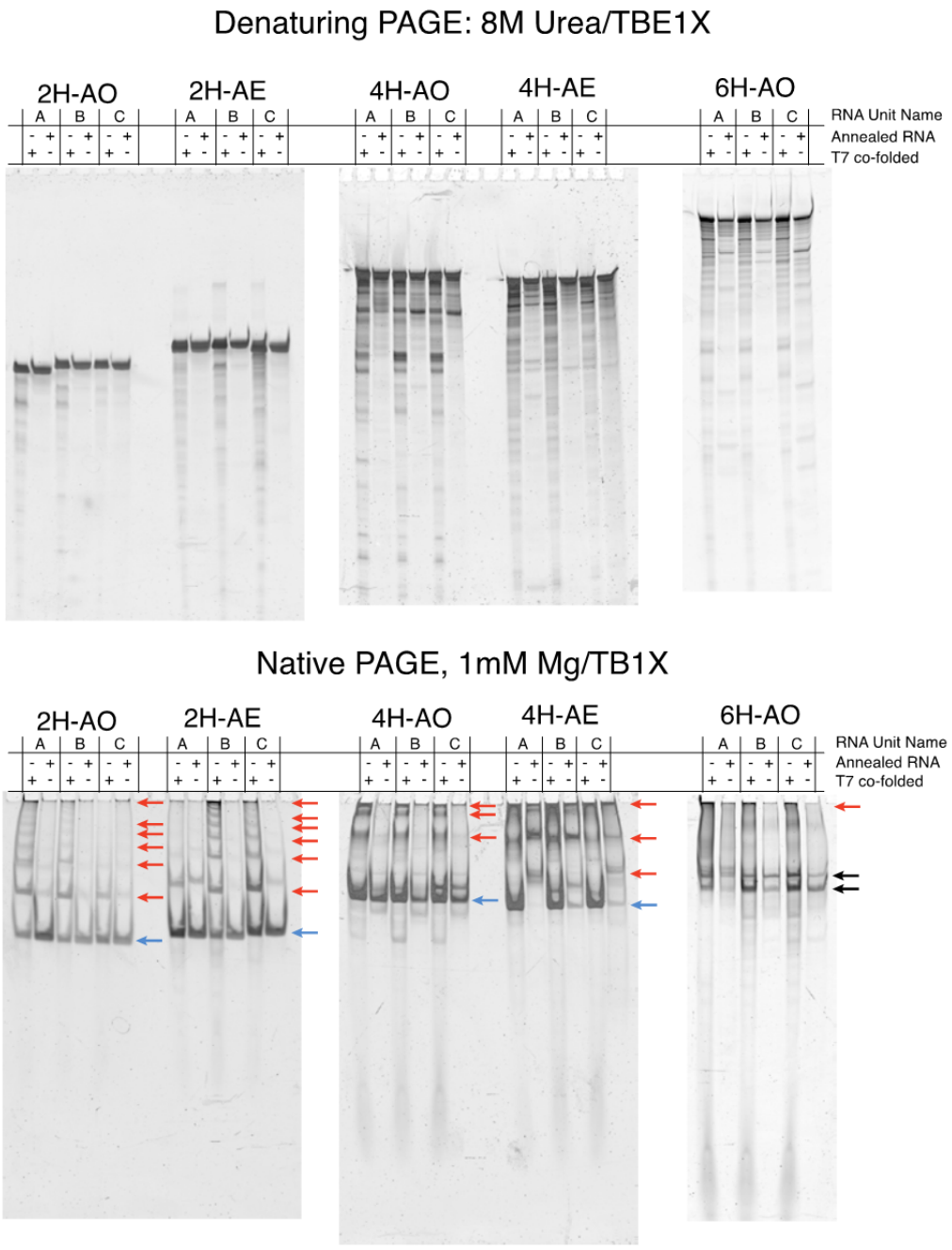


Fig. S13. PAGE gels.

(top) Denaturing 6% Acrylamide:bis (19:1) 8M Urea/TBE 1X PAGE gel, >50°C. (bottom) Native 6% Acrylamide:bis (29:1) 1mM $Mg(OAc)_2$/TB 1X PAGE gel, 4°C. Lanes with gel purified/37°C-incubated RNA tiles are shown adjacent to co-transcriptionally prepared tiles. The desired products are indicated (blue arrows) on the native PAGE, when we were able to identify them. For the 6H-AO native PAGE there are multiple primary folding products (black arrows). Undesired, potentially kinetic products are marked in the native PAGE (red arrows).

Fig. S13 (caption continued).
Under denaturing PAGE, more aborted transcripts are clearly seen in the cotranscriptional samples; the 4H-AO/AE gel is slightly oversaturated such that full-length tile bands appear light, but minor truncation products can be clearly seen. Under native PAGE, higher-order structures are seen for all tiles, but the amount varies from tile to tile and preparation method. For example, a larger amount of higher molecular weight (undesired) structures are seen cotranscriptionally for 4H-AO tiles, whereas solution-incubated samples have a greater fraction of monomers. In contrast, for 4H-AE tiles, solution-incubated samples have almost no monomer bands and instead seem dominated by one of a couple different products, potentially kinetic products due to the introduction of salt at room temperature. Higher molecular weight products are expected to be multimers which form via undesired intermolecular binding of the intramolecular 180° KLs, rather than nonspecific binding of the 120° KLs.

For 6H-AO tiles both cotranscriptionally folding and solution incubation the majority of material seems to run in one of two relatively closely space bands, which seem unlikely to be monomer and dimer bands. Rather we suggest that one band is correctly folded monomer and the other a distorted structure due to the opposing curvature of the two domains of the 6H-AO tile.

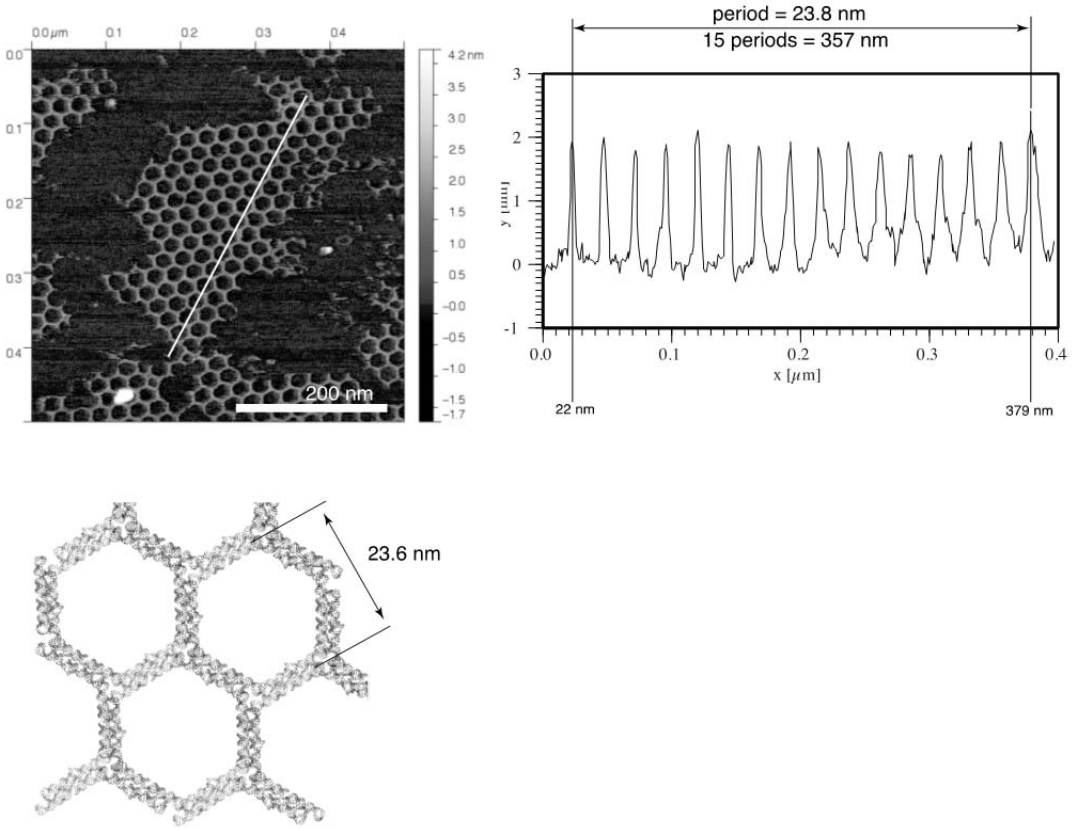


Fig. S14. AFM tile-size measurement (2H-AE-ST, mica annealing).

AFM height profile analysis of 2H-AE single-tile lattices prepared by the mica-annealing protocol.

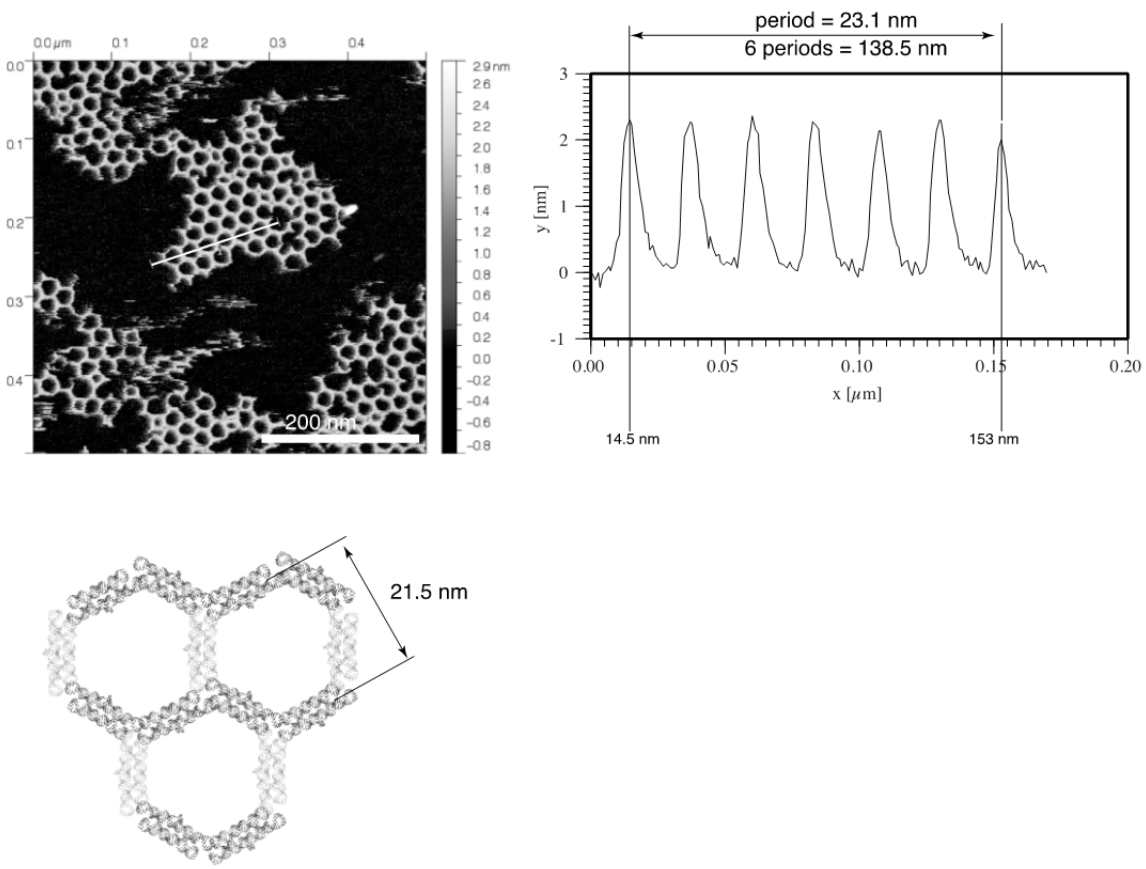


Fig. S15. AFM tile-size measurement (2H-AO-ST, mica anealing).

AFM height profile analysis of 2H-AO single-tile lattices prepared by the mica-annealing protocol.

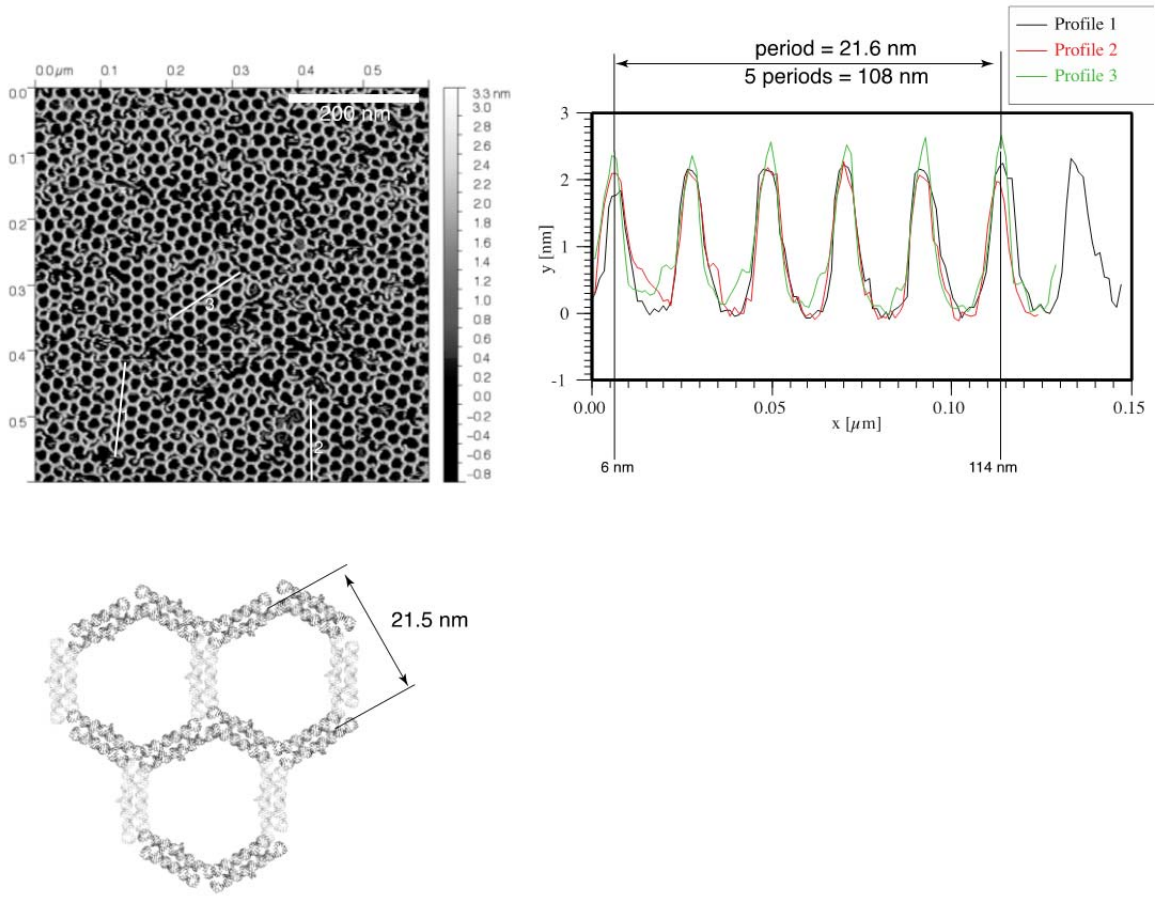


Fig. S16. AFM tile-size measurement (2H-AO-A/B/C, mica annealing).

AFM height profile analysis of 2H-AO three-tile lattices prepared by the mica-annealing protocol.

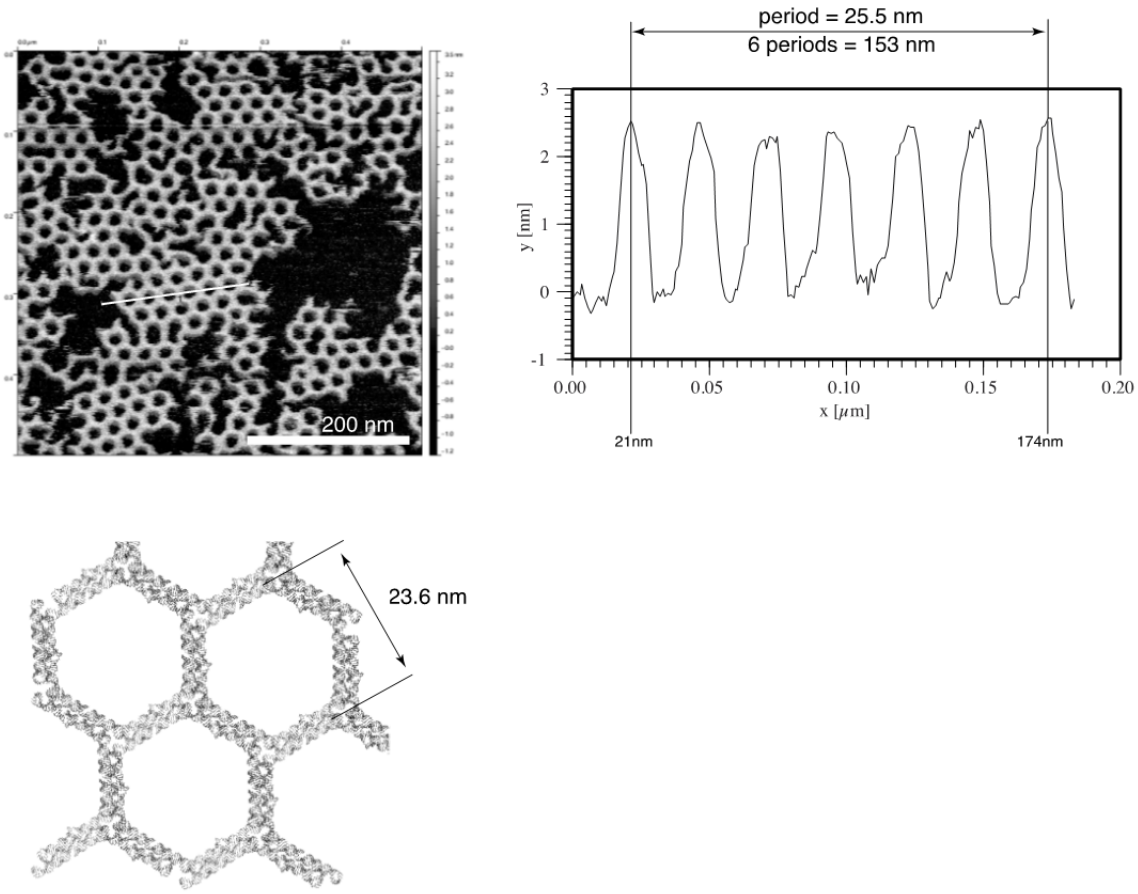


Fig. S17. AFM tile-size measurement (2H-AE-A/B/C, mica-annealing).

AFM height profile analysis of 2H-AE three-tile lattices prepared by the mica-annealing protocol.

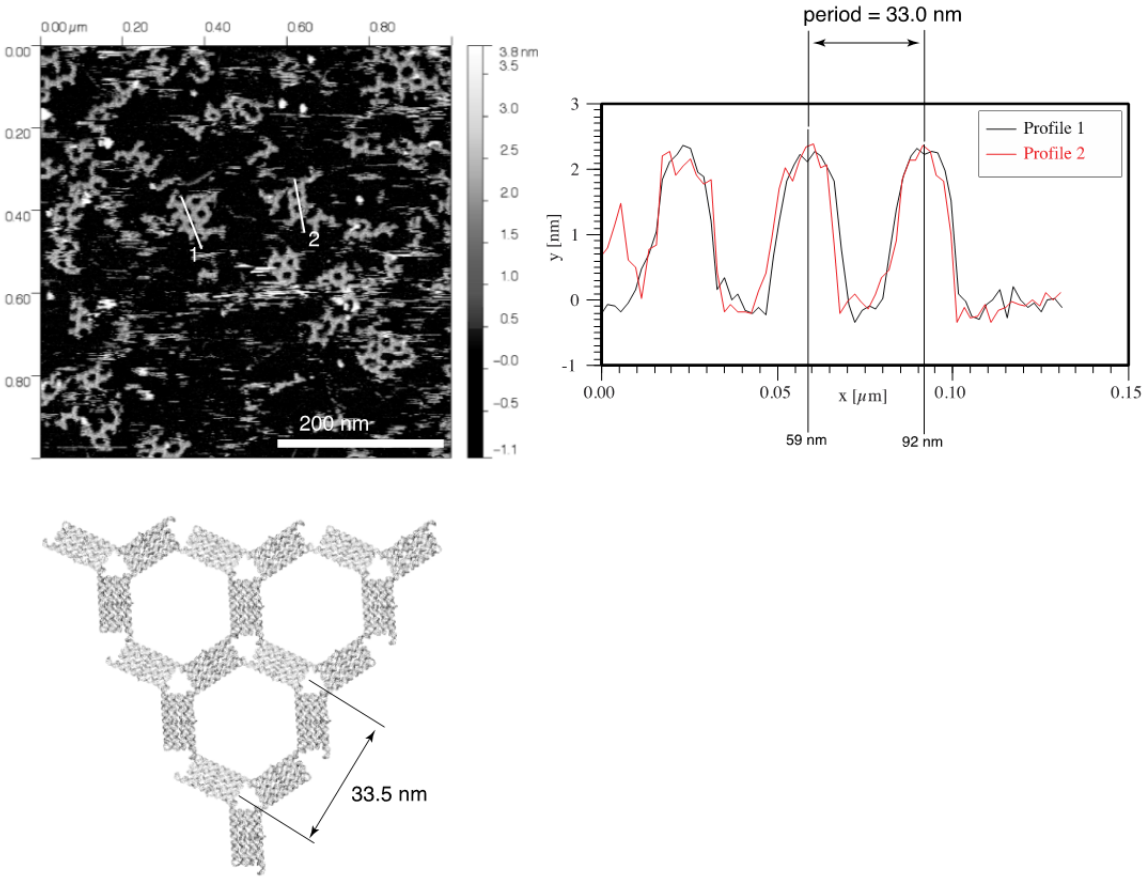


Fig. S18. AFM tile-size measurement (4H-AE-A/B/C, cotranscriptional assembly). AFM height profile analysis of 4H-AE three-tile assembly prepared by the cotranscriptional assembly protocol.

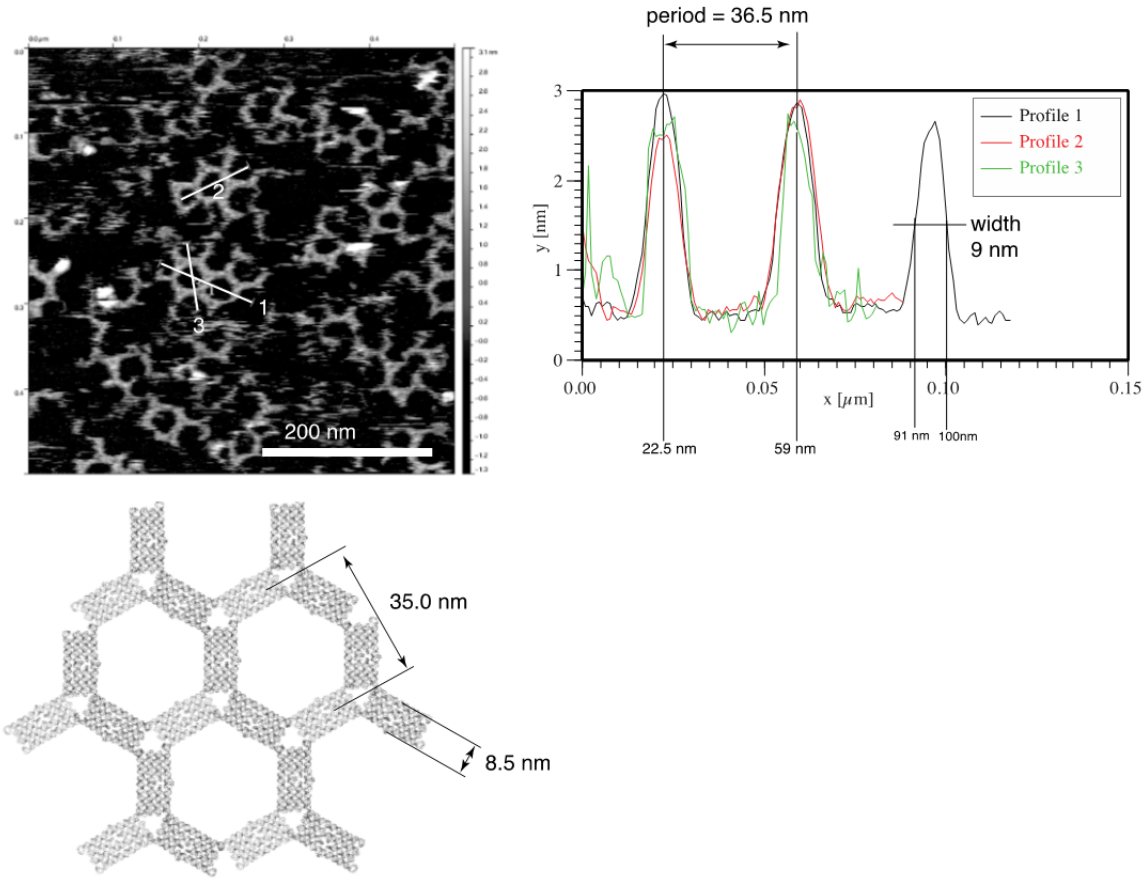


Fig. S19. AFM tile-size measurement (4H-AO-A/B/C, mica annealing).
AFM height profile analysis of 4H-AO three-tile lattices prepared by the mica annealing protocol.

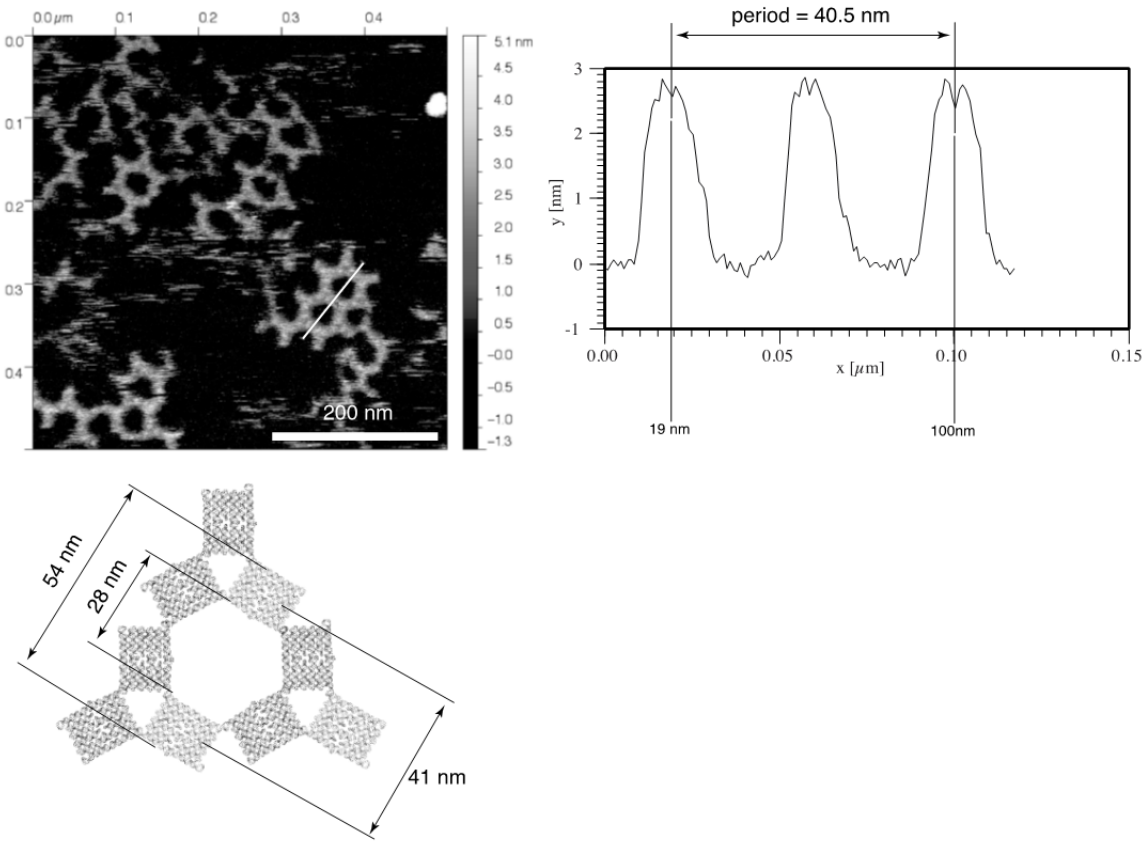


Fig. S20. AFM tile-size measurement (6H-AO-A/B/C, mica annealing).

AFM height profile analysis of 6H-AO three-tile lattices prepared by the mica annealing protocol.

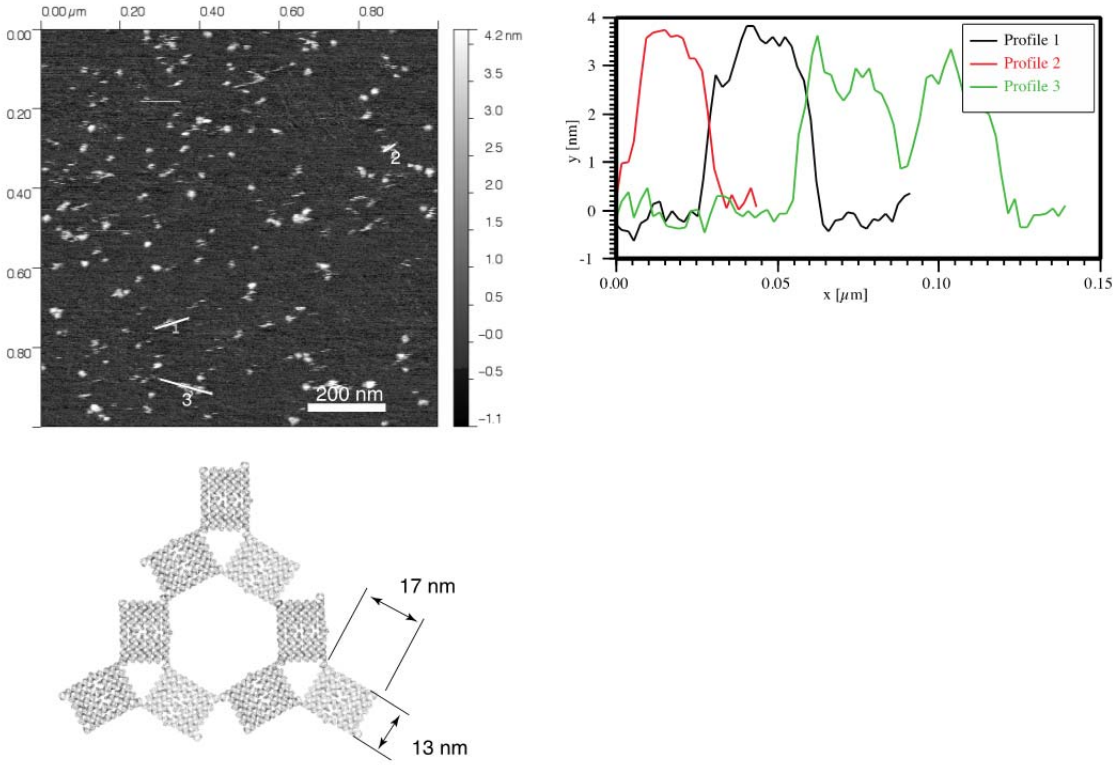


Fig. S21. AFM tile-size measurement (6H-AO-A/B/C, cotranscriptional assembly).
AFM height profile analysis of 6H-AO three-tile lattices prepared by the cotranscriptional assembly protocol.

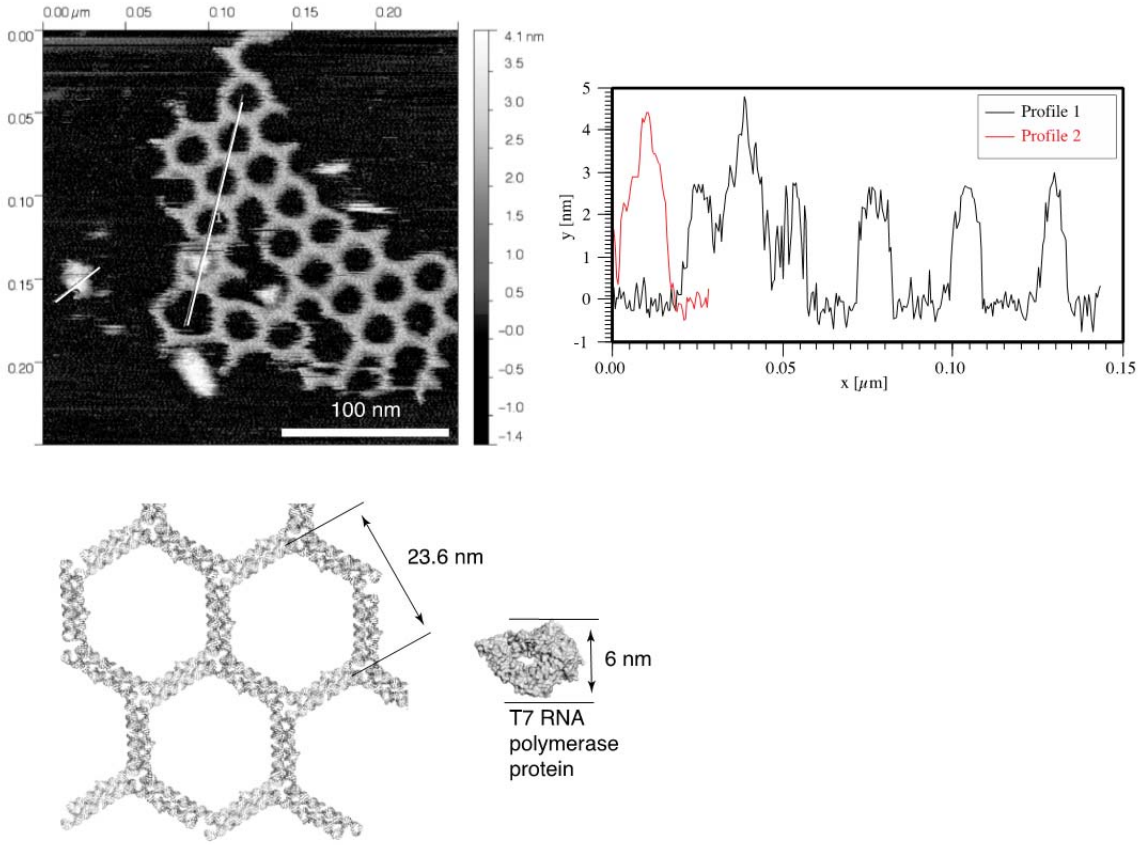


Fig. S22. AFM tile-size measurement (2H-AE-ST, cotranscriptional assembly / mica annealing).

AFM height profile analysis of 2H-AE single-tile lattices prepared by the cotranscriptional assembly / mica annealing protocol.

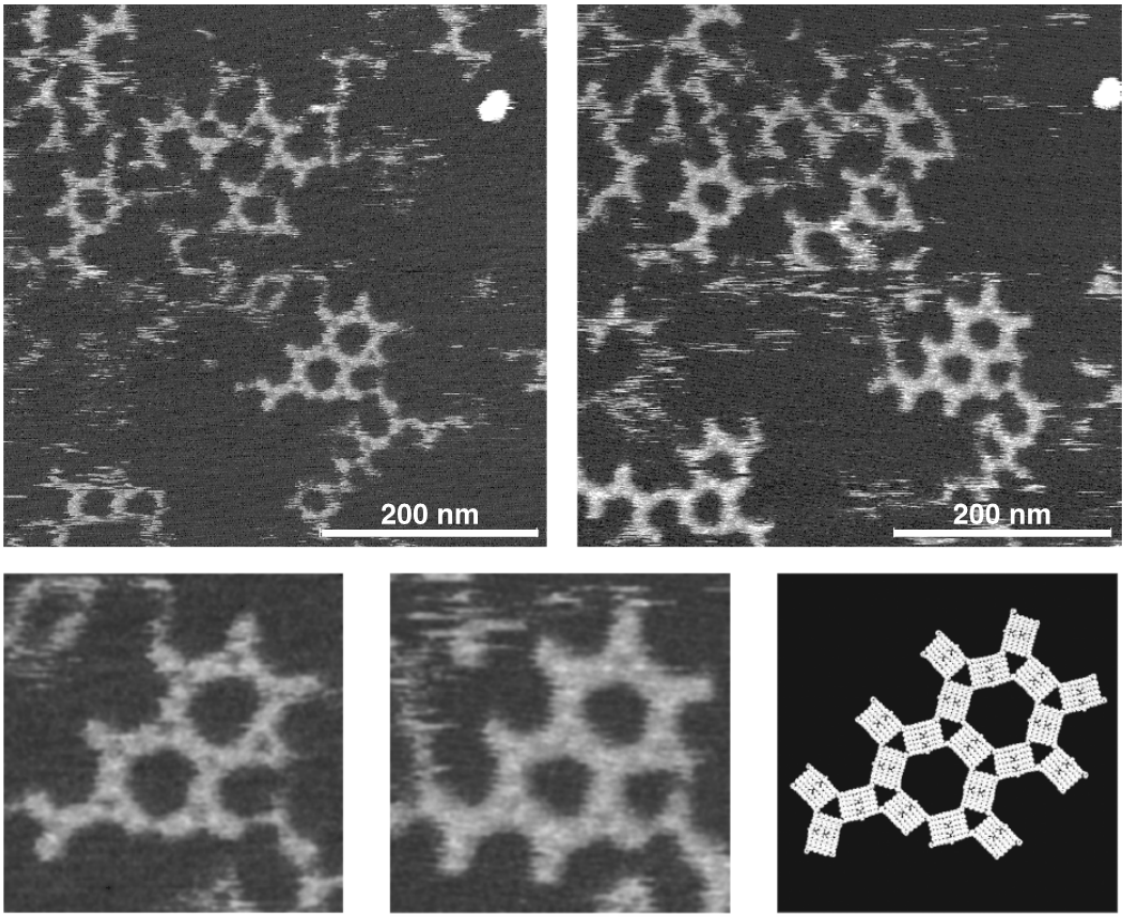


Fig. S23. 6H-AO-A/B/C hexagonal lattice detail.

Full-scale AFM views of the 6H-AO three-tile lattices prepared by the mica-annealing protocol. Two interpolated zoom-ins are shown at bottom, along with a 3D computer model for comparison.

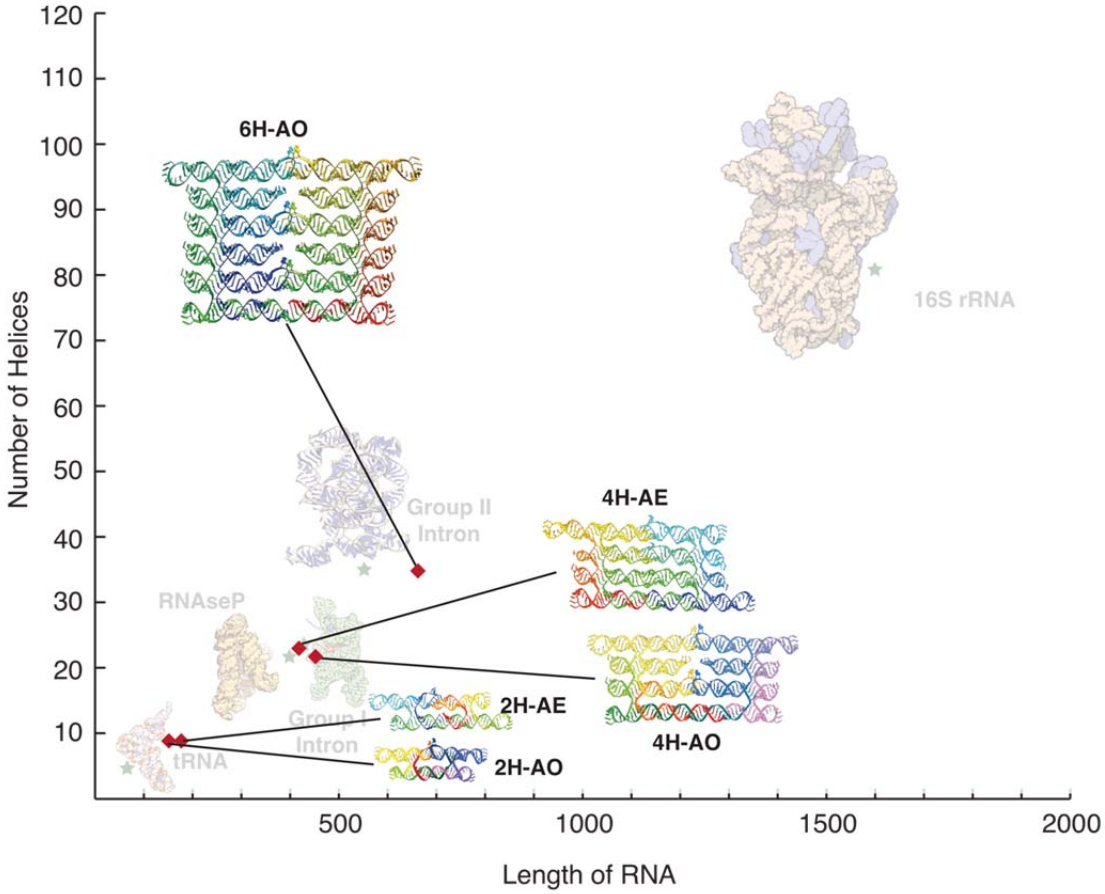


Fig. S24. Complexity of natural and artificial RNA structures.

The complexity of an RNA structure is naturally measured by the number of different helical subdomains (*4*) that must form during folding, since the formation of each helical subdomain involves a separate binding event. Helical subdomains can be quite small, for example each 2 or 3 nucleotide section of a dovetail seam is considered a separate helical subdomain. Here we plot helical subdomains as a function of the number of nucleotides in an RNA structure (length). Natural RNAs are shown in the background for comparison to the artificial RNA tiles generated in this study. By this measure, the 6H-AO tile, is of similar structural complexity to the Group II intron. We note that all structures graphed have similar complexity per nucleotide, with helical subdomains averaging roughly 15-20 nucleotides in length.

Table S1. Sequence and structure for RNA tiles.

Sequence Name	RNA Sequence
2H-AO-A	GGAACGAUCCUGAAGGAGGCACGGGAACCGGGUCCGCAGGCUGGACCCGGUACGGUCGCGU UCUGACCGUUCGUUCCGCUGAGGUCGGUGAAGCCUCCACGCCGAGCUCCGGGUGAUGGUUA CCCGGAGCGCUCCCGUCCACUGGGAGCCCUCAGC
2H-AO-B	GGAAGUCCACUGAAGCUCGCACGGUGACCUCCAUCAGAACGCGAUGGAGGUUCCGGGACCA CGACCCGGAGACUUCCGCAGAGCCCGUUGAAGCGAGCACGACGGUCGGACCGCGAGACGUG CGGUCCGAGCAGGUAACCAUCACCUGCGCUCUGC
2H-AO-C	GGAACGGUCAUGAAGAGCCUACGUGAGACCCGUGGUCGUGGUCCACGGGUCGACUCCAGCC UGCGGAGUCCCGUUCCGCCUAGUCACCUGAAAGGCUCACGGGUGCCUGUCGGUAGUGGACA CCGACAGGUCGUUCACGUCUCGAACGAACUAGGC
2H-AO-SC	GGAAUGAACCUGAAGGAGGCACGGGUACGGGUCGCGAGACGUGCGACCCGUUCAGGCACGU CUCGCCUGAUCAUUCCGCACAGGUCGGUGAAGCCUCCACGCCGAGCUCGACCCGUCCACUG GGUCGAGCGCCUACAGUGGACGUAGGCCCUGUGC
2H-AO-dotbracket	((((((((((((..[[[[[[.)))))((((((((.......))))))))))((((((.... ...))))))))))))((((((((((((..]]]]]].))))))(((((((((.......)))))))))((((((.......)))))))))))))
2H-AE-A	GGAACGCGUGCCAGGGACCGCGUCCACUGCGGUUCCUGAGUGUGAUGGUUACACUGGACCU GAAGCCUCCACGGGUCCGCACGUGUUCCGCAGGGCGAGGUCGGUGAAGGAGGCACGCCGAA CGUCUGACCGCAGGCUGGUCGGACGUACUGCGCGUUCUGCAGUCCUCGUCCUGC
2H-AE-B	GGAACGAGUACGCCGUAGGGUAACCAUCACCCUGCGGCACUCCGAGACGUGGAGUUCUGGU GAAGCGAGCACGCCAGAGUACUUGUUCCGCCAGGACGGAACGGUGAAGCUCGCACGCCGUC UCGUGCCGUAGAACGCACGGUACGAGCCUAUACCACGAAUAGGUCCGUUCUGGC
2H-AE-C	GGAACGGAGGCCCAGGGAUCCACGUCUCGGAUCUCUGGUCGGCAGUGGACGCCGAGGUACU GAAAGGCUCACGGUACCGCCUCUGUUCCGCUCCGGAGCCUCCGUGAAGAGCCUACGCGGAU CACCUGAGGUCGUGGUCCUCGGGUGAGCACCAGCCUGCGGUGCGGCUCUGGAGC
2H-AE-SC	GGAAUUCAGGAGCUCGGGAGCGUCCACUGCUCCUGAGCGCUCCAGUGGACGGAGCCCUGGU GAAGCCUCCACGCCAGGUCCUGGAUUCCGCAUGUAUCGAGGACUGAAGGAGGCACGGUCCC ACGGGCUCCGAGACGUGGAGUCCGUGUCCGCACGUCUCGCGGAUCGAUGCAUGC
2H-AE-dotbracket	((((((((((((((((((((.......))))))))))))(((((.......)))))))))) (..[[[[[[.)))))))))))))))))))(((((((((((((((..]]]]]].))))))) (((((((((.......)))))))))))(((((.......))))))))))))))))
2H-AE-RECT-A	GGAACUGAGUGGCCUGAAGUCACCACGGGCGCUCGUGAAGUCCACACGCGAGCCACGGUGA AGCCUCCACGCCGUGCACUCGGUUCCGCCUGUCGCUGUCACUGAAGGAGGCACGGUGAGGA CUGAAGCGAGCACGGUCCGACUGGUGAAGAACGCACGCCAGUCCAGCGGCAGGC
2H-AE-RECT-A-dotbracket	((((((((((((((((.........)))))))(((((((.........)))))))))))))) .[[[[[[.)))))))))))))))))))(((((((((((((((..]]]]]].))))))))) (((.........)))))))(((((((((.........))))))))))))))))))
2H-AE-RECT-B	GGAACGCUGUGGCACUGAAGCUCGCACGGUGCGCAGCUGAAGCGUUCACGGCUGCCCGUCU GAAGCCUCCACGGACGGCACAGUGUUCCGCUCGUCCUGAUCGGUGAAGGAGGCACGCCGAG AGCUGAAGGUGACACGGCUCCAGCCUGAAGUGGACACGGGCUGUCAGGGCGAGC
2H-AE-RECT-B-dotbracket	((((((((((((((((.........)))))))(((((((.........)))))))))))))) (..[[[[[[.)))))))))))))))))))(((((((((((((((..]]]]]].))))))) (((((.........)))))))(((((((.........))))))))))))))))))

Table S1. (continued)

Sequence Name	RNA Sequence
4H-AO-A	GGAACUUAUACGGAGCCAGCUCGGCCAAUUGACUGAAGGAGGCACGGUCAGUUGGCCC ACGAUGAGUCCAUUCGUAAGAAUGGGCUCAUCCCUACGCGUGUGCCUGAAGUCCACAC GGGCACAUGCGUAGGCAGUGGAGUGCAGGCUACUCUACUGGUGCAGACUACGGUCUGG AGGCCACUACGGUGGCGCACCUGCGUUCUAGGUGCCUGGUUCCGUGUAAGUUCCGCUC CCGGUGAUCAGGCACGUCCGUCUAGGUGAAGCCUCCACGCCUAGGCGGACCUCGCGUC AUCCUGGCGAGAGCCAGGGUGACGCCAUUCCGGCAGCUCUGAAGUGGACACGGAGCUG UCGGAAUGCAGUCUGGUGAUGGUUACCGGACUGGAGCUCCAUUCGUGGAGGUGGCAGC UACGGCUGCGCGAGUGUCCACUACUCGCCCUGGUCAUCGGGAGC
4H-AO-B	GGAACUUAUACGGAGCCAGCUCGGCCAAUUGACUGAAGGAGGCACGGUCAGUUGGCCC ACGAUGAGUCCAUUCGUAAGAAUGGGCUCAUCCCUACGCGUGUGCCUGAAGUCCACAC GGGCACAUGCGUAGGCAGUGGAGUAGAACGCACUCUACUGGUGCAGACUACGGUCUGG AGGCCACUACGGUGGCGCACCUACCACGAAGGUGCCUGGUUCCGUGUAAGUUCCGCUC CCGGUGAUCAGGCACGUCCGUCUAGGUGAAGCCUCCACGCCUAGGCGGACCUCGCGUC AUCCUGGCGAGAGCCAGGGUGACGCCAUUCCGGCAGCUCUGAAGUGGACACGGAGCUG UCGGAAUGCAGUCUGGUGAGACGUACCGGACUGGAGCUCCAUUCGUGGAGGUGGCAGC UACGGCUGCGCGAGUAACCAUCACUCGCCCUGGUCAUCGGGAGC
4H-AO-C	GGAACUUAUACGGAGCCAGCUCGGCCAAUUGACUGAAGGAGGCACGGUCAGUUGGCCC ACGAUGAGUCCAUUCGUAAGAAUGGGCUCAUCCCUACGCGUGUGCCUGAAGUCCACAC GGGCACAUGCGUAGGCAGUGGAGUUCGUGGUACUCUACUGGUGCAGACUACGGUCUGG AGGCCACUACGGUGGCGCACCUAGCCUGCAGGUGCCUGGUUCCGUGUAAGUUCCGCUC CCGGUGAUCAGGCACGUCCGUCUAGGUGAAGCCUCCACGCCUAGGCGGACCUCGCGUC AUCCUGGCGAGAGCCAGGGUGACGCCAUUCCGGCAGCUCUGAAGUGGACACGGAGCUG UCGGAAUGCAGUCUGGUAGUGGACACCGGACUGGAGCUCCAUUCGUGGAGGUGGCAGC UACGGCUGCGCGAGUACGUCUCACUCGCCCUGGUCAUCGGGAGC
4H-AO-dotbracket	((((((((((((((((((((((((((((((((((..[[[[[[.))))))))))))))))))(((((((((((((((....))))))))))))))(((((((((((((((..[[[[[[.)))))))))))))))))(((((((((.......)))))))))))(((((....))))))))(((((....)))))(((((((.......)))))))))))))))))))))))))))(((((((((((((((((((((((((((((((..]]]]]].)))))))))))))))))))))))) (((((((....)))))))))))))))(((((((((((((((..]]]]]].)))))))))))))))))(((((((((.......)))))))))))))(((((....)))))))))))(((....)))))(((((((.......))))))))))))))))))))

Table S1. (continued)

Sequence Name	RNA Sequence
4H-AE-A	GGAAGUGAGUAGUAGUCCACCGUCUACGUCCACUGUAGGCGGUGCGCUCACUUCGGUG AGGGCCAGCUACGGCUGGGACCCGAUGGUUGGGUCGACGGAGGCUGAAGCGAGCACGG CCUCUGUCCCGAGCUCUGCUGAAAGGCUCACGGCAGGGCUCCGGCAGGUGGCUGAAGC CUCCACGGCCAUCUGCGACUGCUACUCGCUUCCGCGAAAUGUCAAUACGCGCAGCGAC GGUGAAGGAGGCACGCCGUUGCUGGGCACUGAGGGUGAAGAGCCUACGCCCUCGGUGG CGAAGCGACCUGAAGCUCGCACGGGUCGUUUCGCGCAGGGUCAGGCAGGCUCUGAUCC UGCCCGAUCCUACGGGAUCCGGAACAUACGUGUUCGGACCGGCGUUCUCGGUCCCGUA UUGGCAUUUCGC
4H-AE-B	GGAAGUGAGUAGUAGUCCACCGUCUACAACCAUCGUAGGCGGUGCGCUCACUUCGGUG AGGGCCAGCUACGGCUGGGACCCGAGACGUGGGUCGACGGAGGCUGAAGCGAGCACGG CCUCUGUCCCGAGCUCUGCUGAAAGGCUCACGGCAGGGCUCCGGCAGGUGGCUGAAGC CUCCACGGCCAUCUGCGACUGCUACUCGCUUCCGCGAAAUGUCAAUACGCGCAGCGAC GGUGAAGGAGGCACGCCGUUGCUGGGCACUGAGGGUGAAGAGCCUACGCCCUCGGUGG CGAAGCGACCUGAAGCUCGCACGGGUCGUUUCGCGCAGGGUCAGAGAACGCCUGAUCC UGCCCGAUCCUACGGGAUCCGGAACAUACGUGUUCGGACCGACCACGACGGUCCCGUA UUGGCAUUUCGC
4H-AE-C	GGAAGUGAGUAGUAGUCCACCGUCUACACGUCUCGUAGGCGGUGCGCUCACUUCGGUG AGGGCCAGCUACGGCUGGGACCCAGUGGACGGGUCGACGGAGGCUGAAGCGAGCACGG CCUCUGUCCCGAGCUCUGCUGAAAGGCUCACGGCAGGGCUCCGGCAGGUGGCUGAAGC CUCCACGGCCAUCUGCGACUGCUACUCGCUUCCGCGAAAUGUCAAUACGCGCAGCGAC GGUGAAGGAGGCACGCCGUUGCUGGGCACUGAGGGUGAAGAGCCUACGCCCUCGGUGG CGAAGCGACCUGAAGCUCGCACGGGUCGUUUCGCGCAGGGUCAGUCGUGGUCUGAUCC UGCCCGAUCCUACGGGAUCCGGAACAUACGUGUUCGGACCGAGCCUGCCGGUCCCGUA UUGGCAUUUCGC
4H-AE-dotbracket	((((((((((((((((((((((((((.....))))))))))))))))))....)))))))))))....))))))))))(((((.....)))))((((((((((..[[[[[.)))))))))))))((((((((((..[[[[[.))))))))))))))((((((((((..[[[[[[.)) ((((..]]]]].))))))))))))(((((((((((..]]]]].)))))))))))) ((((((((((..]]]]].))))))))))))))(((((((((.....)))))))))))))((((....))))))))((((....))))))((((.....))))))))))))))))))))))

Table S1. (continued)

Sequence Name	RNA Sequence
6H-AO-A	GGAAUCUGACGAAGUCGUGCUCGGUGCUGCACCUGAAGGAGGCACGGGUGCGGCACCC UCGUACGCAUAGUCAGUAAUGACUGUGCGUACCUCCAAUUCAAGGCUGAAGCUCGCAC GGCCUUGGAUUGCGCGUCGGCGACCUUGGGAACAAGGUUGCCGACCCGCUAUAGGCAC CUGAAGUCCACACGGGUGCUUAUAGCGGCUCGGGCUCGCAGGCUGAGCCUGAGGCGGU GACUUCGGUCACGAGCGGGAUUCGUCCCGGAGCGACAUUCGUGUCGGAGGCAGCUACG GCUGCGCCUUCGCGUUCUGAAGGCCACGAUUUCGUCGGAUUCCGUCUUGCAGAGGUCG GCUCGACUUAAUGUCUGAAGCCUCCACGGACAUUGAGUCCUCGCGCUCUUACAUGGUG ACAUGUAGGAGCGCCACCGUCUGUUGACUGAAGCGAGCACGGUCAAUAGACGCACCUC AGCUAAUACUGUAAAGUAUUGGCUGAGCUCAGGAUAUCUGGUGAAGUGGACACGCCAG AUGUCCUGAGGUCGUACGUGAUGGUUACGUGCGACGUGGAACCUACGGGUUCGUGCAC GAUUCGUCGUGGAGCAGCCUACGGGCUGGAGGCUCCUACGGGAGCGCACGCGUCCACU GCGUGCCCGACUUCUGCAGGAC
6H-AO-B	GGAAUCUGACGAAGUCGUGCUCGGUGCUGCACCUGAAGGAGGCACGGGUGCGGCACCC UCGUACGCAUAGUCAGUAAUGACUGUGCGUACCUCCAAUUCAAGGCUGAAGCUCGCAC GGCCUUGGAUUGCGCGUCGGCGACCUUGGGAACAAGGUUGCCGACCCGCUAUAGGCAC CUGAAGUCCACACGGGUGCUUAUAGCGGCUCGGGCUCAGAACGCGAGCCUGAGGCGGU GACUUCGGUCACGAGCGGGAUUCGUCCCGGAGCGACAUUCGUGUCGGAGGCAGCUACG GCUGCGCCUUCACCACGAGAAGGCCACGAUUUCGUCGGAUUCCGUCUUGCAGAGGUCG GCUCGACUUAAUGUCUGAAGCCUCCACGGACAUUGAGUCCUCGCGCUCUUACAUGGUG ACAUGUAGGAGCGCCACCGUCUGUUGACUGAAGCGAGCACGGUCAAUAGACGCACCUC AGCUAAUACUGUAAAGUAUUGGCUGAGCUCAGGAUAUCUGGUGAAGUGGACACGCCAG AUGUCCUGAGGUCGUACGUGAGACGUACGUGCGACGUGGAACCUACGGGUUCGUGCAC GAUUCGUCGUGGAGCAGCCUACGGGCUGGAGGCUCCUACGGGAGCGCACGCAACCAUC GCGUGCCCGACUUCUGCAGGAC
6H-AO-C	GGAAUCUGACGAAGUCGUGCUCGGUGCUGCACCUGAAGGAGGCACGGGUGCGGCACCC UCGUACGCAUAGUCAGUAAUGACUGUGCGUACCUCCAAUUCAAGGCUGAAGCUCGCAC GGCCUUGGAUUGCGCGUCGGCGACCUUGGGAACAAGGUUGCCGACCCGCUAUAGGCAC CUGAAGUCCACACGGGUGCUUAUAGCGGCUCGGGCUCUCGUGGUGAGCCUGAGGCGGU GACUUCGGUCACGAGCGGGAUUCGUCCCGGAGCGACAUUCGUGUCGGAGGCAGCUACG GCUGCGCCUUCAGCCUGCGAAGGCCACGAUUUCGUCGGAUUCCGUCUUGCAGAGGUCG GCUCGACUUAAUGUCUGAAGCCUCCACGGACAUUGAGUCCUCGCGCUCUUACAUGGUG ACAUGUAGGAGCGCCACCGUCUGUUGACUGAAGCGAGCACGGUCAAUAGACGCACCUC AGCUAAUACUGUAAAGUAUUGGCUGAGCUCAGGAUAUCUGGUGAAGUGGACACGCCAG AUGUCCUGAGGUCGUACGUAGUGGACACGUGCGACGUGGAACCUACGGGUUCGUGCAC GAUUCGUCGUGGAGCAGCCUACGGGCUGGAGGCUCCUACGGGAGCGCACGCACGUCUC GCGUGCCCGACUUCUGCAGGAC
6H-AO-dotbracket	((((((((((((((((((((((((((((((((((((..[[[[[[.)))))))))))))))))))) ((((((((((((((....))))))))))))))((((((((((((((..[[[[[[.)))))))))))))))((((((((((((((....))))))))))))))(((((((((((((((..[[[[[[.))))))))))))))))))((((((((.......))))))))))))(((((....)))))))))(((((....)))))))))(((((....)))))))))(((((....)))))((((((.......))))))))))))))))))))))))))))))(((((((((((((((((((((((((((((..]]]]]].))))))))))))))))(((((((((((((... .))))))))))))))(((((((((((((((..]]]]]].))))))))))))))((((((((((((((((....))))))))))))))(((((((((((((((..]]]]]].))))))))))))))))(((((((((.......)))))))))))))(((((....)))))))))(((((....)))))))))(((((....)))))))))(((((....)))))(((((((.......)))))))))))))))))))

Table S2. Modeled and measured tile dimensions.

Tile Class	Size	Tile set	Assembly type	Modeled Spacing	Measured Spacing
2H-AO	156nt	3 tiles	annealed	21.5 nm	21.6 nm
2H-AO	156nt	1 tile	annealed	21.5 nm	23.1 nm
2H-AE	176nt	3 tiles	annealed	23.6 nm	25.5 nm
2H-AE	176nt	1 tile	annealed	23.6 nm	23.8 nm
4H-AE	418nt	3 tiles	T7 co-trans	33.5 nm	33.0 nm
4H-AO	450nt	3 tiles	annealed	35.0 nm	36.5 nm
6H-AO	660nt	3 tiles	annealed	41.0 nm	40.5 nm

Note S1. DNA Templates and Primers for PCR.

Forward Primer for all designs:

>T7_GGAA.FWD
TTCTAATACGACTCACTATAGGAA

Reverse Primers and dsDNA Templates:

2H-AO 1-tile System:

>2H-AO-ST.REV
GCACAGGGCCTACGTCC
>2H-AO-SC.TMP
GCACAGGGCCTACGTCCACTGTAGGCGCTCGACCCAGTGGACGGGTCGAGCTCGGCGTGGAGGCTTCACCGACCTGTGCGGAATGATCA
GGCGAGACGTGCCTGAACGGGTCGCACGTCTCGCGACCCGTACCCGTGCCTCCTTCAGGTTCATTCCTATAGTGAGTCGTATTAGAA

2H-AO 3-tile System:

>2H-AO-A.REV
GCTGAGGGCTCCCAGTGG
>2H-AO-A.TMP
GCTGAGGGCTCCCAGTGGACGGGAGCGCTCCGGGTAACCATCACCCGGAGCTCGGCGTGGAGGCTTCACCGACCTCAGCGGAACGAACG
GTCAGAACGCGACCGTACCGGGTCCAGCCTGCGGACCCGGTTCCCGTGCCTCCTTCAGGATCGTTCCTATAGTGAGTCGTATTAGAA

>2H-AO-B.REV
GCAGAGCGCAGGTGATGG
>2H-AO-B.TMP
GCAGAGCGCAGGTGATGGTTACCTGCTCGGACCGCACGTCTCGCGGTCCGACCGTCGTGCTCGCTTCAACGGGCTCTGCGGAAGTCTCC
GGGTCGTGGTCCCGGAACCTCCATCGCGTTCTGATGGAGGTCACCGTGCGAGCTTCAGTGGACTTCCTATAGTGAGTCGTATTAGAA

>2H-AO-C.REV
GCCTAGTTCGTTCGAGACG
>2H-AO-C.TMP
GCCTAGTTCGTTCGAGACGTGAACGACCTGTCGGTGTCCACTACCGACAGGCACCCGTGAGCCTTTCAGGTGACTAGGCGGAACGGGAC
TCCGCAGGCTGGAGTCGACCCGTGGACCACGACCACGGGTCTCACGTAGGCTCTTCATGACCGTTCCTATAGTGAGTCGTATTAGAA

2H-AE 1-tile system:

>2H-AE-ST.REV
GCATGCATCGATCCGCG
>2H-AE-SC.TMP
GCATGCATCGATCCGCGAGACGTGCGGACACGGACTCCACGTCTCGGAGCCCGTGGGACCGTGCCTCCTTCAGTCCTCGATACATGCGG
AATCCAGGACCTGGCGTGGAGGCTTCACCAGGGCTCCGTCCACTGGAGCGCTCAGGAGCAGTGGACGCTCCCGAGCTCCTGAATTCCTA
TAGTGAGTCGTATTAGAA

2H-AE 3-tile system:

>2H-AE-A.REV
GCACGCCCTGGGCATCAG
>2H-AE-A.TMP
GCACGCCCTGGGCATCAGAACGCGATGCGGGTCAGGACAGCCTGCGTCCCGACCCTCGGCGTGCCTCCTTCACCGACCAGGACGTGCGG
AACAGCTTGGCTGGCGTGGAGGCTTCACCAGCGTCTGAACCATCCAGACTCGGATCACCAGTGGACGGTGACCCGACAAGCCGTTCCTA
TAGTGAGTCGTATTAGAA

>2H-AE-B.REV
GCCAGAACGGACCTATTCGTG
>2H-AE-B.TMP

GCCAGAACGGACCTATTCGTGGTATAGGCTCGTACCGTGCGTTCTACGGCACGAGACGGCGTGCGAGCTTCACCGTTCCGTCCTGGCGG
AACAAGTACTCTGGCGTGCTCGCTTCACCAGAACTCCACGTCTCGGAGTGCCGCAGGGTGATGGTTACCCTACGGCGTACTCGTTCCTA
TAGTGAGTCGTATTAGAA

>2H-AE-C.REV
GCTCCAGAGCCGCACC
>2H-AE-C.TMP
GCTCCAGAGCCGCACCGCAGGCTGGTGCTCACCCGAGGACCACGACCTCAGGTGATCCGCGTAGGCTCTTCACGGAGGCTCCGGAGCGG
AACAGAGGCGGTACCGTGAGCCTTTCAGTACCTCGGCGTCCACTGCCGACCAGAGATCCGAGACGTGGATCCCTGGGCCTCCGTTCCTA
TAGTGAGTCGTATTAGAA

2H-AE Rectalinaer tiles:

>2H-AE-RECT-A.REV
GCCTGCCGCTGGACTGG
>2H-AE-RECT-A.TMP
GCCTGCCGCTGGACTGGCGTGCGTTCTTCACCAGTCGGACCGTGCTCGCTTCAGTCCTCACCGTGCCTCCTTCAGTGACAGCGACAGGC
GGAACCGAGTGCACGGCGTGGAGGCTTCACCGTGGCTCGCGTGTGGACTTCACGAGCGCCCGTGGTGACTTCAGGCCACTCAGTTCCTA
TAGTGAGTCGTATTAGAA

>2H-AE-RECT-B.REV
GCTCGCCCTGACAGCC
>2H-AE-RECT-B.TMP
GCTCGCCCTGACAGCCCGTGTCCACTTCAGGCTGGAGCCGTGTCACCTTCAGCTCTCGGCGTGCCTCCTTCACCGATCAGGACGAGCGG
AACACTGTGCCGTCCGTGGAGGCTTCAGACGGGCAGCCGTGAACGCTTCAGCTGCGCACCGTGCGAGCTTCAGTGCCACAGCGTTCCTA
TAGTGAGTCGTATTAGAA

4H-AE 3-tile system:

>4H-AE-ABC.REV
GCGAAATGCCAATACGGGACC

>4H-AE-A.TMP
GCGAAATGCCAATACGGGACCGAGAACGCCGGTCCGAACACGTATGTTCCGGATCCCGTAGGATCGGGCAGGATCAGAGCCTGCCTGAC
CCTGCGCGAAACGACCCGTGCGAGCTTCAGGTCGCTTCGCCACCGAGGGCGTAGGCTCTTCACCCTCAGTGCCCAGCAACGGCGTGCCT
CCTTCACCGTCGCTGCGCGTATTGACATTTCGCGGAAGCGAGTAGCAGTCGCAGATGGCCGTGGAGGCTTCAGCCACCTGCCGGAGCCC
TGCCGTGAGCCTTTCAGCAGAGCTCGGGACAGAGGCCGTGCTCGCTTCAGCCTCCGTCGACCCAACCATCGGGTCCCAGCCGTAGCTGG
CCCTCACCGAAGTGAGCGCACCGCCTACAGTGGACGTAGACGGTGGACTACTACTCACTTCCTATAGTGAGTCGTATTAGAA

>4H-AE-B.TMP
GCGAAATGCCAATACGGGACCGTCGTGGTCGGTCCGAACACGTATGTTCCGGATCCCGTAGGATCGGGCAGGATCAGGCGTTCTCTGAC
CCTGCGCGAAACGACCCGTGCGAGCTTCAGGTCGCTTCGCCACCGAGGGCGTAGGCTCTTCACCCTCAGTGCCCAGCAACGGCGTGCCT
CCTTCACCGTCGCTGCGCGTATTGACATTTCGCGGAAGCGAGTAGCAGTCGCAGATGGCCGTGGAGGCTTCAGCCACCTGCCGGAGCCC
TGCCGTGAGCCTTTCAGCAGAGCTCGGGACAGAGGCCGTGCTCGCTTCAGCCTCCGTCGACCCACGTCTCGGGTCCCAGCCGTAGCTGG
CCCTCACCGAAGTGAGCGCACCGCCTACGATGGTTGTAGACGGTGGACTACTACTCACTTCCTATAGTGAGTCGTATTAGAA

>4H-AE-B.TMP
GCGAAATGCCAATACGGGACCGGCAGGCTCGGTCCGAACACGTATGTTCCGGATCCCGTAGGATCGGGCAGGATCAGACCACGACTGAC
CCTGCGCGAAACGACCCGTGCGAGCTTCAGGTCGCTTCGCCACCGAGGGCGTAGGCTCTTCACCCTCAGTGCCCAGCAACGGCGTGCCT
CCTTCACCGTCGCTGCGCGTATTGACATTTCGCGGAAGCGAGTAGCAGTCGCAGATGGCCGTGGAGGCTTCAGCCACCTGCCGGAGCCC
TGCCGTGAGCCTTTCAGCAGAGCTCGGGACAGAGGCCGTGCTCGCTTCAGCCTCCGTCGACCCGTCCACTGGGTCCCAGCCGTAGCTGG
CCCTCACCGAAGTGAGCGCACCGCCTACGAGACGTGTAGACGGTGGACTACTACTCACTTCCTATAGTGAGTCGTATTAGAA

4H-AO 3-tile system:

>4H-AO-ABC.REV
GCTCCCGATGACCAGGGC

>4H-AO-A.TMP
GCTCCCGATGACCAGGGCGAGTAGTGGACACTCGCGCAGCCGTAGCTGCCACCTCCACGAATGGAGCTCCAGTCCGGTAACCATCACCA
GACTGCATTCCGACAGCTCCGTGCTCGCTTCAGAGCTGCCGGAATGGCGTCACCCTGGCTCTCGCCAGGATGACGCGAGGTCCGCCTAG
GCGTGGAGGCTTCACCTAGACGGACGTGCCTGATCACCGGGAGCGGAACTTACACGGAACCAGGCACCTAGAACGCAGGTGCGCCACCG
TAGTGGCCTCCAGACCGTAGTCTGCACCAGTAGAGTAGCCTGCACTCCACTGCCTACGCATGTGCCCGTGCGAGCTTCAGGCACACGCG
TAGGGATGAGCCCATTCTTACGAATGGACTCATCGTGGGCCAACTGACCGTGCCTCCTTCAGTCAATTGGCCGAGCTGGCTCCGTATAA
GTTCCTATAGTGAGTCGTATTAGAA

>4H-AO-B.TMP
GCTCCCGATGACCAGGGCGAGTGATGGTTACTCGCGCAGCCGTAGCTGCCACCTCCACGAATGGAGCTCCAGTCCGGTACGTCTCACCA
GACTGCATTCCGACAGCTCCGTGCTCGCTTCAGAGCTGCCGGAATGGCGTCACCCTGGCTCTCGCCAGGATGACGCGAGGTCCGCCTAG
GCGTGGAGGCTTCACCTAGACGGACGTGCCTGATCACCGGGAGCGGAACTTACACGGAACCAGGCACCTTCGTGGTAGGTGCGCCACCG
TAGTGGCCTCCAGACCGTAGTCTGCACCAGTAGAGTGCGTTCTACTCCACTGCCTACGCATGTGCCCGTGCGAGCTTCAGGCACACGCG
TAGGGATGAGCCCATTCTTACGAATGGACTCATCGTGGGCCAACTGACCGTGCCTCCTTCAGTCAATTGGCCGAGCTGGCTCCGTATAA
GTTCCTATAGTGAGTCGTATTAGAA

>4H-AO-C.TMP
GCTCCCGATGACCAGGGCGAGTGAGACGTACTCGCGCAGCCGTAGCTGCCACCTCCACGAATGGAGCTCCAGTCCGGTGTCCACTACCA
GACTGCATTCCGACAGCTCCGTGCTCGCTTCAGAGCTGCCGGAATGGCGTCACCCTGGCTCTCGCCAGGATGACGCGAGGTCCGCCTAG
GCGTGGAGGCTTCACCTAGACGGACGTGCCTGATCACCGGGAGCGGAACTTACACGGAACCAGGCACCTGCAGGCTAGGTGCGCCACCG
TAGTGGCCTCCAGACCGTAGTCTGCACCAGTAGAGTACCACGAACTCCACTGCCTACGCATGTGCCCGTGCGAGCTTCAGGCACACGCG
TAGGGATGAGCCCATTCTTACGAATGGACTCATCGTGGGCCAACTGACCGTGCCTCCTTCAGTCAATTGGCCGAGCTGGCTCCGTATAA
GTTCCTATAGTGAGTCGTATTAGAA

6H-AO 3-tile system:

>6H-AO-ABC.REV
GTCCTGCAGAAGTCGGGCACGC

>6H-AO-A.TMP
GCAGAAGTCGGGCACGCAGTGGACGCGTGCGCTCCCGTAGGAGCCTCCAGCCCGTAGGCTGCTCCACGACGAATCGTGCACGAACCCGT
AGGTTCCACGTCGCACGTAACCATCACGTACGACCTCAGGACATCTGGCGTGTCCACTTCACCAGATATCCTGAGCTCAGCCAATACTT
TACAGTATTAGCTGAGGTGCGTCTATTGACCGTGCTCGCTTCAGTCAACAGACGGTGGCGCTCCTACATGTCACCATGTAAGAGCGCGA
GGACTCAATGTCCGTGGAGGCTTCAGACATTAAGTCGAGCCGACCTCTGCAAGACGGAATCCGACGAAATCGTGGCCTTCAGAACGCGA
AGGCGCAGCCGTAGCTGCCTCCGACACGAATGTCGCTCCGGGACGAATCCCGCTCGTGACCGAAGTCACCGCCTCAGGCTCAGCCTGCG
AGCCCGAGCCGCTATAAGCACCCGTGTGGACTTCAGGTGCCTATAGCGGGTCGGCAACCTTGTTCCCAAGGTCGCCGACGCGCAATCCA
AGGCCGTGCGAGCTTCAGCCTTGAATTGGAGGTACGCACAGTCATTACTGACTATGCGTACGAGGGTGCCGCACCCGTGCCTCCTTCAG
GTGCAGCACCGAGCACGACTTCGTCAGATTCCTATAGTGAGTCGTATTAGAA

>6H-AO-B.TMP
GCAGAAGTCGGGCACGCGATGGTTGCGTGCGCTCCCGTAGGAGCCTCCAGCCCGTAGGCTGCTCCACGACGAATCGTGCACGAACCCGT
AGGTTCCACGTCGCACGTACGTCTCACGTACGACCTCAGGACATCTGGCGTGTCCACTTCACCAGATATCCTGAGCTCAGCCAATACTT
TACAGTATTAGCTGAGGTGCGTCTATTGACCGTGCTCGCTTCAGTCAACAGACGGTGGCGCTCCTACATGTCACCATGTAAGAGCGCGA
GGACTCAATGTCCGTGGAGGCTTCAGACATTAAGTCGAGCCGACCTCTGCAAGACGGAATCCGACGAAATCGTGGCCTTCTCGTGGTGA
AGGCGCAGCCGTAGCTGCCTCCGACACGAATGTCGCTCCGGGACGAATCCCGCTCGTGACCGAAGTCACCGCCTCAGGCTCGCGTTCTG
AGCCCGAGCCGCTATAAGCACCCGTGTGGACTTCAGGTGCCTATAGCGGGTCGGCAACCTTGTTCCCAAGGTCGCCGACGCGCAATCCA
AGGCCGTGCGAGCTTCAGCCTTGAATTGGAGGTACGCACAGTCATTACTGACTATGCGTACGAGGGTGCCGCACCCGTGCCTCCTTCAG
GTGCAGCACCGAGCACGACTTCGTCAGATTCCTATAGTGAGTCGTATTAGAA

>6H-AO-C.TMP
GCAGAAGTCGGGCACGCGAGACGTGCGTGCGCTCCCGTAGGAGCCTCCAGCCCGTAGGCTGCTCCACGACGAATCGTGCACGAACCCGT
AGGTTCCACGTCGCACGTGTCCACTACGTACGACCTCAGGACATCTGGCGTGTCCACTTCACCAGATATCCTGAGCTCAGCCAATACTT
TACAGTATTAGCTGAGGTGCGTCTATTGACCGTGCTCGCTTCAGTCAACAGACGGTGGCGCTCCTACATGTCACCATGTAAGAGCGCGA
GGACTCAATGTCCGTGGAGGCTTCAGACATTAAGTCGAGCCGACCTCTGCAAGACGGAATCCGACGAAATCGTGGCCTTCGCAGGCTGA
AGGCGCAGCCGTAGCTGCCTCCGACACGAATGTCGCTCCGGGACGAATCCCGCTCGTGACCGAAGTCACCGCCTCAGGCTCACCACGAG
AGCCCGAGCCGCTATAAGCACCCGTGTGGACTTCAGGTGCCTATAGCGGGTCGGCAACCTTGTTCCCAAGGTCGCCGACGCGCAATCCA
AGGCCGTGCGAGCTTCAGCCTTGAATTGGAGGTACGCACAGTCATTACTGACTATGCGTACGAGGGTGCCGCACCCGTGCCTCCTTCAG
GTGCAGCACCGAGCACGACTTCGTCAGATTCCTATAGTGAGTCGTATTAGAA

Note 2. Perl script for tracing text-based RNA design files.

The RNA-trace script was written in Perl to easily convert a text-based RNA design file into dot-bracket or similar format. The text-based RNA design file was developed to allow easy editing and copy-pasting of structural elements (see examples in Fig. S5 and 6). It uses unicode characters (╭, ╰, ╮, ╯, │, ┼, ┆, *) to define the strand path and base pairs, the numbers 5 and 3 to define the 5' and 3' ends, and standard sequence constraints (C, G, U, A, R, Y, K, N). The script finds the 5' end in the file and traces its way along the strand path until it finds the 3' end. Along the way it notes the sequence and base-pairs connections and prints out the result. In its current implementation it outputs NUPACK design format for easy use in this particular program. To run the program on the command line type:

```
> perl RNA-trace.pl design.txt > nupack.txt
```

The program enters an alternative mode if you add a second input file. It reads the second input file as a sequence file and threads it onto the design file to create a blueprint containing the primary and secondary structure (examples in Fig. S7). To run the program in this mode on the command line type:

```
> perl RNA-trace.pl design.txt sequence.txt > blueprint.txt
```

Below is a copy of the current code that you are welcome to use or modify as you like:

```
#!/usr/bin/env perl

#  -*- perl -*-

# RNA-trace
# written by Ebbe S. Andersen <esa@inano.au.dk>, 2013

use strict;
use warnings FATAL => qw ( all );
use Data::Dumper;
use Encode;
use utf8;

# >>>>>>>>>>>>>>>>>> RUN PROGRAM <<<<<<<<<<<<<<<<<<<<

my ( $file1, $file2, $line, @cols, $seq );

( $file1, $file2 ) = @ARGV;

my @pri = ( );
if ( defined $file2 ) {
    open FILE, "< $file2";
    while ( $line = <FILE> ) {
        print "$line";
        print "\n";
        @pri = split(//, $line);
    }
} else {
    $file2 = "no";
}

if ( not open FILE, "< $file1" ) { die "file not found!"; }

# translate unicode to symbols:
# ---------------------------------------
# symbol unicode       unique description
# ---------------------------------------
# _      \302\240      240    space
# -      \342\224\200  200    straight
# i      \342\224\202  202    up/down connection
# p      \342\224\212  212    pair
# x      \342\224\274  274    strand path cross
# L      \342\225\255  260    bottom left corner
# r      \342\225\260  255    top left corner
```

```
# 7      \342\225\257  256    top right corner
# J      \342\225\256  257    bottom right corner
# ----------------------------------------

my @m = ( );
my @n = ( );
my @t = ( );
@cols = ( );
my $i = 0;
my $j = 0;
my $cols = 0;
my $l = 0;
my $k = 0;
while ( $line = <FILE> ) {
    $l = length $line;
    $k = 0;
    for ($i=0;$i<$l;$i++) {
        $cols = substr("$line", $i, 1);
        if ( $cols eq " " ) { $m[$j][$k] = " "; $k++; }        # space
        if ( $cols eq "*" ) { $m[$j][$k] = "*"; $k++; }        # space
        if ( $cols eq "\257" ) { $m[$j][$k] = "J"; $k++; }     # left up
        if ( $cols eq "\200" ) { $m[$j][$k] = "-"; $k++; }     # straight
        if ( $cols eq "\260" ) { $m[$j][$k] = "L"; $k++; }     # right up
        if ( $cols eq "\255" ) { $m[$j][$k] = "r"; $k++; }     # right down
        if ( $cols eq "\256" ) { $m[$j][$k] = "7"; $k++; }     # left down
        if ( $cols eq "\202" ) { $m[$j][$k] = "i"; $k++; }     # down
        if ( $cols eq "\212" ) { $m[$j][$k] = "p"; $k++; }     # pair
        if ( $cols eq "I"    ) { $m[$j][$k] = "I"; $k++; }     # non-WC
        if ( $cols eq "\274" ) { $m[$j][$k] = "x"; $k++; }     # cross
        if ( $cols =~ /\w/ ) { $m[$j][$k] = "$cols"; $k++; }
        if ( $cols =~ /\n/ ) { $m[$j][$k] = "$cols"; $k++; }
    }
    $j++;
}

# print back translation

print "Input file:\n";
for ($i=0;$i<1000;$i++) {
    for ($j=0;$j<1000;$j++) {
        if ( defined $m[$i][$j] ) {
            if ( $m[$i][$j] =~ /[NGACURYK35\*]/ ) { print "$m[$i][$j]"; }     # space
            if ( $m[$i][$j] eq "\n" ){ print "$m[$i][$j]"; }     # space
            if ( $m[$i][$j] eq " " ) { print "$m[$i][$j]"; }     # space
            if ( $m[$i][$j] eq "7" ) { print "\342\225\256"; }   # left up
            if ( $m[$i][$j] eq "-" ) { print "\342\224\200"; }   # straight
            if ( $m[$i][$j] eq "r" ) { print "\342\225\255"; }   # right up
            if ( $m[$i][$j] eq "L" ) { print "\342\225\260"; }   # right down
            if ( $m[$i][$j] eq "J" ) { print "\342\225\257"; }   # left down
            if ( $m[$i][$j] eq "i" ) { print "\342\224\202"; }   # down
            if ( $m[$i][$j] eq "p" ) { print "\342\224\212"; }   # pair
            if ( $m[$i][$j] eq "I" ) { print "I"; }              # non-WC
            if ( $m[$i][$j] eq "x" ) { print "\342\224\274"; }   # cross
        }
    }
}

# Find 5 prime end

my $r = 0;
my $c = 0;
my $d = "left";
for ($i=0;$i<1000;$i++) {
    for ($j=0;$j<1000;$j++) {
        if ( defined $m[$i][$j] ) {
            if ($m[$i][$j] =~ /\d+/ ) {
                if ( scalar $m[$i][$j] && $m[$i][$j] == 5 ) {
                    $r = $i;
                    $c = $j;
                    if ( $m[$r][$c+1] =~ /[NGCAURYK-]/ ) { $d = "right"; }
                    if ( $m[$r][$c-1] =~ /[NGCAURYK-]/ ) { $d = "left"; }
                    print "\nThe 5p end is found at row $r, column $c and is running $d.\n";
                }
            }
        }
    }
}

# Find total amount of nucleotides in blueprint

my $nt = 0;
for ($i=0;$i<1000;$i++) {
    for ($j=0;$j<1000;$j++) {
        if ( defined $m[$i][$j] ) {
            if ($m[$i][$j] =~ /[NGCAURYK]/ ) {
                $nt++;
            }
        }
```

```
    }
}
print "There are $nt nucleotides in the blueprint file.\n";

# Now trace the structure

my $r2 = scalar ($r);
my $c2 = scalar ($c);
my $d2 = $d;
my $num = 0;
my @seq = ( );
my $test = "test";
for ($k=0;$k<$nt+10000;$k++) {
    # TRACE HORIZONTAL STRAND
    if ( $file2 eq "no" ) {
        if ( $d eq "right" && $m[$r][$c+1] =~ /[xNGCAURYK-]/ ) {
            if ( $m[$r][$c+1] =~ /[NGCAURYK]/ ) { $num++; $n[$r][$c+1] = $num; push @seq, $m[$r][$c+1]; }
            $c++;
        }
        if ( $d eq "left"  && $m[$r][$c-1] =~ /[xNGCAURYK-]/ ) {
            if ( $m[$r][$c-1] =~ /[NGCAURYK]/ ) { $num++; $n[$r][$c-1] = $num; push @seq, $m[$r][$c-1]; }
            $c--;
        }
    } else { # when a sequence is available as file2 then we add it on the m-grid here
        if ( $d eq "right" && $m[$r][$c+1] =~ /[xNGCAURYK-]/ ) {
            if ( $m[$r][$c+1] =~ /[NGCAURYK]/ ) { $m[$r][$c+1] = $pri[$num]; $num++; $n[$r][$c+1] = $num; }
            $c++;
        }
        if ( $d eq "left"  && $m[$r][$c-1] =~ /[xNGCAURYK-]/ ) {
            if ( $m[$r][$c-1] =~ /[NGCAURYK]/ ) { $m[$r][$c-1] = $pri[$num]; $num++; $n[$r][$c-1] = $num; }
            $c--;
        }
    }
    # CROSS-OVER
    if ( $d eq "right" ) {
        if ( $m[$r][$c+1] eq "7" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r+$i][$c+1] eq "J" ) { $c++; $d = "left"; $r = $r + $i; last; }
                if ( $m[$r+$i][$c+1] eq "L" ) { $c++; $d = "right"; $r = $r + $i; last; }
            }
        }
        if ( $m[$r][$c+1] eq "J" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r-$i][$c+1] eq "7" ) { $c++; $d = "left"; $r = $r - $i; last; }
                if ( $m[$r-$i][$c+1] eq "r" ) { $c++; $d = "right"; $r = $r - $i; last; }
            }
        }
    }
    if ( $d eq "left" ) {
        if ( $m[$r][$c-1] eq "L" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r-$i][$c-1] eq "7" ) { $c--; $d = "left"; $r = $r - $i; last; }
                if ( $m[$r-$i][$c-1] eq "r" ) { $c--; $d = "right"; $r = $r - $i; last; }
            }
        }
        if ( $m[$r][$c-1] eq "r" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r+$i][$c-1] eq "J" ) { $c--; $d = "left"; $r = $r + $i; last; }
                if ( $m[$r+$i][$c-1] eq "L" ) { $c--; $d = "right"; $r = $r + $i; last; }
            }
        }
    }
    if ( $m[$r][$c+1] =~ /\d/ ) { if ( $m[$r][$c+1] == 3 ) { $test = "success"; last; } }
    if ( $m[$r][$c-1] =~ /\d/ ) { if ( $m[$r][$c-1] == 3 ) { $test = "success"; last; } }
}
if ( $test eq "success" ) {
    print "The structure has been successfully traced (3p end found).\n";
} else {
    print "The trace through the structure failed (3p end not found). Ended at row $r, column $c.\n";
}
print "The are " . scalar (@seq) . " nts from 5p to 3p.\n";

# Find base pairs

$r = scalar ($r2); # reset row
$c = scalar ($c2); # reset column
$d = $d2; # restore 5p direction
my @a = ( );
my @b = ( );
my @p = ( );
for ($k=0;$k<$nt+10000;$k++) {
    # TRACE HORIZONTAL STRAND
    if ( $d eq "right" && $m[$r][$c+1] =~ /[xNGCAURYK-]/ ) {
        if ( $m[$r][$c+1] =~ /[NGCAURYK]/ ) {
            if ( $m[$r+1][$c+1] =~ /[Ip\*]/ ) {
                push @a, $n[$r][$c+1];
                push @b, $n[$r+2][$c+1];
                push @p, $m[$r+1][$c+1];
```

```
            } elsif ( $m[$r-1][$c+1] =~ /[Ip\*]/ ) {
                push @a, $n[$r][$c+1];
                push @b, $n[$r-2][$c+1];
                push @p, $m[$r-1][$c+1];
            } else {
                push @a, $n[$r][$c+1];
                push @b, 0;
                push @p, "-";
            }
        }
        $c++;
    }
    if ( $d eq "left"  && $m[$r][$c-1] =~ /[xNGCAURYK-]/ ) {
        if ( $m[$r][$c-1] =~ /[NGCAURYK]/ ) {
            if ( $m[$r+1][$c-1] =~ /[Ip\*]/ ) {
                push @a, $n[$r][$c-1];
                push @b, $n[$r+2][$c-1];
                push @p, $m[$r+1][$c-1];
            } elsif ( $m[$r-1][$c-1] =~ /[Ip\*]/ ) {
                push @a, $n[$r][$c-1];
                push @b, $n[$r-2][$c-1];
                push @p, $m[$r-1][$c-1];
            } else {
                push @a, $n[$r][$c-1];
                push @b, 0;
                push @p, "-";
            }
        }
        $c--;
    }
    if ( $d eq "right" && $m[$r][$c+1] =~ /[x-]/ ) { $c++; }
    if ( $d eq "left"  && $m[$r][$c-1] =~ /[x-]/ ) { $c--; }
    # CROSS-OVER
    if ( $d eq "right" ) {
        if ( $m[$r][$c+1] eq "7" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r+$i][$c+1] eq "J" ) { $c++; $d = "left"; $r = $r + $i; last; }
                if ( $m[$r+$i][$c+1] eq "L" ) { $c++; $d = "right"; $r = $r + $i; last; }
            }
        }
        if ( $m[$r][$c+1] eq "J" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r-$i][$c+1] eq "7" ) { $c++; $d = "left"; $r = $r - $i; last; }
                if ( $m[$r-$i][$c+1] eq "r" ) { $c++; $d = "right"; $r = $r - $i; last; }
            }
        }
    }
    if ( $d eq "left" ) {
        if ( $m[$r][$c-1] eq "L" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r-$i][$c-1] eq "7" ) { $c--; $d = "left"; $r = $r - $i; last; }
                if ( $m[$r-$i][$c-1] eq "r" ) { $c--; $d = "right"; $r = $r - $i; last; }
            }
        }
        if ( $m[$r][$c-1] eq "r" ) {
            for ( $i=1; $i<=100; $i++ ) {
                if ( $m[$r+$i][$c-1] eq "J" ) { $c--; $d = "left"; $r = $r + $i; last; }
                if ( $m[$r+$i][$c-1] eq "L" ) { $c--; $d = "right"; $r = $r + $i; last; }
            }
        }
    }
}

# print sequence

my $pri = 0;
if ( $file2 eq "no" ) {
    print "\n\nOutput: The sequence and structure\n\n";
    foreach $seq ( @seq ) {
        if ( $seq =~ /[NGCAURYK]/ ) { print "$seq"; }
    }
    print "\n";
} else {
    print "\n\nOutput: The sequence and structure\n\n";
    foreach $pri ( @pri ) {
        if ( $pri =~ /[NGCAURYK]/ ) { print "$pri"; }
    }
    print "\n";
}

# print structure

my $a = 0;
my $b = 0;
$i = 0;
foreach $a ( @a ) {
    if ( defined $b[$i] && defined $a ) {
        if ( $a > $b[$i] and $b[$i] != 0 ) {
```

```
            if ( $p[$i] eq "p" ) { print ")"; }
            if ( $p[$i] eq "I" ) { print ">"; }
            if ( $p[$i] eq "*" ) { print "]"; }
        }
        if ( $a < $b[$i] and $b[$i] != 0 ) {
            if ( $p[$i] eq "p" ) { print "("; }
            if ( $p[$i] eq "I" ) { print "<"; }
            if ( $p[$i] eq "*" ) { print "["; }
        }
        if ( $b[$i] == 0 ) { print "."; }
    } else {
        print ".";
    }
    $i++;
}
print "\n";

# print blueprint

if ( $file2 ne "no" ) {
    print "\n\nOutput: 2D diagram with sequence\n";
    for ($i=0;$i<1000;$i++) {
        for ($j=0;$j<1000;$j++) {
            if ( defined $m[$i][$j] ) {
                if ( $m[$i][$j] =~ /[NGACURYK35\*]/ ) { print "$m[$i][$j]"; } # space
                if ( $m[$i][$j] eq "\n" ) { print "$m[$i][$j]"; }    # space
                if ( $m[$i][$j] eq " " ) { print "$m[$i][$j]"; }     # space
                if ( $m[$i][$j] eq "7" ) { print "\342\225\256"; }   # left up
                if ( $m[$i][$j] eq "-" ) { print "\342\224\200"; }   # straight
                if ( $m[$i][$j] eq "r" ) { print "\342\225\255"; }   # right up
                if ( $m[$i][$j] eq "L" ) { print "\342\225\260"; }   # right down
                if ( $m[$i][$j] eq "J" ) { print "\342\225\257"; }   # left down
                if ( $m[$i][$j] eq "i" ) { print "\342\224\202"; }   # down
                if ( $m[$i][$j] eq "p" ) { print "\342\224\212"; }   # pair
                if ( $m[$i][$j] eq "I" ) { print "I"; }              # non-WC
                if ( $m[$i][$j] eq "x" ) { print "\342\224\274"; }   # cross
            }
        }
    }
}
print "\n\nBegin NuPack Code (copy below this line):\n";
print "material = RNA";
print "\ntemperature[C] = 37.0";
print "\ntrials = 4";
print "\nsodium[M] = 1.0";
print "\ndangles = some";
print "\nallowmismatch = true";
print "\nstructure RNA_seq = ";
#put in the 2D constraint here
$a = 0;
$b = 0;
$i = 0;
foreach $a ( @a ) {
    if ( defined $b[$i] && defined $a ) {
        if ( $a > $b[$i] and $b[$i] != 0 ) {
            if ( $p[$i] eq "p" ) { print ")"; }
            if ( $p[$i] eq "I" ) { print "."; }
            if ( $p[$i] eq "*" ) { print "."; }
        }
        if ( $a < $b[$i] and $b[$i] != 0 ) {
            if ( $p[$i] eq "p" ) { print "("; }
            if ( $p[$i] eq "I" ) { print "."; }
        if ( $p[$i] eq "*" ) { print "."; }
        }
        if ( $b[$i] == 0 ) { print "."; }
    } else {
        print ".";
    }
    $i++;
}
print "\ndomain a = ";
$pri = 0;
# print sequence
if ( $file2 eq "no" ) {
    foreach $seq ( @seq ) {
        if ( $seq =~ /[NGCAURYK]/ ) { print "$seq"; }
    }
} else {
    foreach $pri ( @pri ) {
        if ( $pri =~ /[NGCAURYK]/ ) { print "$pri"; }
    }
}
print "\nRNA_seq.seq = a";
print "\nprevent = AAAA, CCCC, GGGG, UUUU, KKKKKK, MMMMMM, RRRRRR, SSSSSS, WWWWWW, YYYYYY\n";
#end NuPack input sections
```